MPC8555E PowerQUICC™ III Integrated Processor Family Reference Manual

Supports:

MPC8555E
MPC8541E

MPC8555ERM
Rev. 2
10/2006

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescale.com

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)
support@freescale.com

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: MPC8555ERM
Rev. 2, 10/2006

Contents

Contents

Chapter 2
Memory Map

Contents

Contents

Contents

Chapter 6
Core Register Summary

Contents

Contents

Chapter 7
L2 Look-Aside Cache/SRAM

Contents

Part III
Memory, Security, and I/O Interfaces

Chapter 8
e500 Coherency Module

Contents

Chapter 9
DDR Memory Controller

Contents

Chapter 10
Programmable Interrupt Controller

Contents

Contents

Chapter 11
I^2C Interface

Contents

Chapter 12
DUART

Contents

Contents

Contents

Contents

Contents

Chapter 14
Three-Speed Ethernet Controllers

Contents

Contents

Contents

Contents

Chapter 15
DMA Controller

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 19
Performance Monitor

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 31
SCC BISYNC Mode

Chapter 32
SCC Transparent Mode

Contents

Chapter 33
SCC AppleTalk Mode

Chapter 34
QUICC Multi-Channel Controller (QMC)

Contents

Contents

Chapter 35
Universal Serial Bus Controller

Contents

Contents

Chapter 37
Fast Communications Controllers (FCCs)

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 42
ATM AAL2

Contents

Chapter 43
Serial Peripheral Interface (SPI)

Contents

Contents

Appendix A
MPC8541E

Appendix B
Revision History

Contents

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

About This Book

This reference manual defines the functionality of the MPC8555E and MPC8541E. It is written from the perspective of the MPC8555E, which is the superset device. For specifics on how to use this manual for the MPC8541E, see Appendix A, "MPC8541E."

The MPC8555E PowerQUICC™ III is a next-generation PowerQUICC™ II integrated communications processor. The MPC8555E provides integration of processing power for networking and communications peripherals, resulting in higher device performance. The MPC8555E contains a PowerPC™ processor core built on Power Architecture™ technology. The e500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the embedded category features of the Power Architecture technology. This book is intended as a companion to the *PowerPC™ e500 Core Family Reference Manual.*

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, "Overview," describes the many features of the MPC8555E integrated communication processor at an overview level. The following chapters are included:

- Chapter 1, "Overview," provides a high-level description of features and functionality of the MPC8555E integrated communication processor. It describes the MPC8555E, its interfaces, and its programming model. The functional operation of the MPC8555E with emphasis on peripheral functions is also described.
- Chapter 2, "Memory Map," describes the memory map of the MPC8555E. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of 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 reset configuration signals (and the modes they define).
- Chapter 4, "Reset, Clocking, and Initialization," describes the hard and soft resets, the power-on reset (POR) sequence, power-on reset configuration, clocking, and initialization of the MPC8555E.

Part II, "e500 Core Complex and L2 Cache," describes the many features of the MPC8555E core processor at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, "Core Complex Overview," provides an overview of the e500 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, "Core Register Summary," provides a listing of the e500 registers in reference form.
- Chapter 7, "L2 Look-Aside Cache/SRAM," describes the L2 cache of the MPC8555E. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

Part III, "Memory, Security, and I/O Interfaces," defines the memory, security and I/O interfaces of the MPC8555E and how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- Chapter 8, "e500 Coherency Module," defines the e500 coherency module and how it facilitates communication between the e500 core complex, the L2 cache, and the other blocks that comprise the coherent memory domain of the MPC8555E.

  The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8555E.
- Chapter 9, "DDR Memory Controller," describes the DDR SDRAM memory controller of the MPC8555E. 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. A large set of special features like DLL software override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, "Programmable Interrupt Controller," describes the embedded programmable interrupt controller (PIC) of the MPC8555E. The PIC is an OpenPIC-compliant interrupt controller that provides interrupt management and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them and delivering them to the CPU for servicing.
- Chapter 11, "I^2C Interface," describes the inter-IC (IIC or I^2C) bus controller of the MPC8555E. 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 MPC8555E powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.
- Chapter 12, "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 13, "Local Bus Controller," describes the MPC8555E local bus controller. 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, regular DRAM devices, extended data output DRAM devices, and other peripherals.

- Chapter 14, "Three-Speed Ethernet Controllers," describes the 2 three-speed Ethernet controllers on the MPC8555E. These controllers provide 10/100/1Gb Ethernet support with a complete set of media-independent interface options including GMII, RGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8555E memory coherency module.
- Chapter 15, "DMA Controller," describes the four-channel general-purpose DMA controller of the MPC8555E. The DMA controller transfers blocks of data independent of the e500 core or external hosts. Data movement occurs among the local address space. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 16, "PCI Bus Interface," describes the MPC8555E PCI controller.
- Chapter 17, "Security Engine (SEC) 2.0," describes the MPC8555E security controller.

Part IV, "Global Functions and Debug," defines other global blocks of the MPC8555E. The following chapters are included:

- Chapter 18, "Global Utilities," defines the global utilities of the MPC8555E. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals
- Chapter 19, "Performance Monitor," describes the performance monitor of the MPC8555E. Note that the MPC8555E performance monitor is similar to but separate from the performance monitor implemented on the e500 core.
- Chapter 20, "Debug Features and Watchpoint Facility," describes the debug features and watchpoint monitor of the MPC8555E.

Part V, "CPM Features," defines the MPC8555E CPM blocks. The following chapters are included:

- Chapter 21, "Communications Processor Module Overview," provides a high-level summary of the MPC8555E features and memory map.
- Chapter 22, "CPM Interrupt Controller," describes the MPC8555E CPM interrupt controller.
- Chapter 23, "Serial Interface with Time-Slot Assigner," describes the MPC8555E serial interface and TSA.
- Chapter 24, "CPM Multiplexing," describes how the CPM multiplexing logic (CMX) connects the physical layer (UTOPIA, MII, modem lines, TDM lines, and proprietary serial lines) to the FCCs and SCCs.
- Chapter 25, "Baud-Rate Generators (BRGs)," describes the eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs and SCCs.
- Chapter 26, "CPM Timers," describes the four identical 16-bit general-purpose CPM timers that can alternately be used as two 32-bit timers.
- Chapter 27, "SDMA Channels," describes the two physical serial DMA (SDMA) channels of the MPC8555E.

- Chapter 28, “Serial Communications Controllers (SCCs),” describes the four serial communications controllers (SCCs) which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks.
- Chapter 29, “SCC UART Mode,” describes how the general SCC mode register (GSMR) is used to configure an SCC channel to function in UART mode, which provides standard serial I/O using asynchronous character-based (start-stop) protocols with RS-232C-type lines.
- Chapter 30, “SCC HDLC Mode,” describes how HDLC mode is selected for an SCC. In HDLC mode, an SCC becomes an HDLC controller, and consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to other SCCs.
- Chapter 31, “SCC BISYNC Mode,” describes how transparent BISYNC mode allows full binary data to be sent with any possible character pattern.
- Chapter 32, “SCC Transparent Mode,” describes how an SCC in transparent mode functions as a high-speed serial-to-parallel and parallel-to-serial converter.
- Chapter 33, “SCC AppleTalk Mode,” describes how the MPC8555E provides LocalTalk protocol support. The AppleTalk controller provides required frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames.
- Chapter 34, “QUICC Multi-Channel Controller (QMC),” describes the QMC protocol, which emulates up to 64 logical channels within one SCC using the same time-division-multiplexed (TDM) physical interface.
- Chapter 35, “Universal Serial Bus Controller,” describes the MPC8555E USB controller, including basic operation, the parameter RAM, and registers.
- Chapter 36, “Serial Management Controllers (SMCs),” describes two serial management controllers, full-duplex ports that can be configured independently to support one of three protocols—UART, transparent, or general-circuit interface (GCI).
- Chapter 37, “Fast Communications Controllers (FCCs),” describes how the FCCs can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, and gateways, and also interface with a wide variety of standard WANs, LANs, and proprietary networks.
- Chapter 38, “FCC HDLC Controller,” describes the MPC8555E FCC HDLC controller.
- Chapter 39, “FCC Transparent Controller,” describes how the FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter.
- Chapter 40, “CPM Fast Ethernet Controller,” describes the fast Ethernet controller in the CPM.
- Chapter 41, “ATM Controller,” describes the ATM controller that provides the ATM and AAL layers of the ATM protocol using the universal test and operations physical layer (PHY) interface for ATM (UTOPIA level II) for both master and slave modes.
- Chapter 42, “ATM AAL2,” describes the AAL2 microcode package.
- Chapter 43, “Serial Peripheral Interface (SPI),” describes the MPC8555E CPM serial peripheral interface (SPI).
- Chapter 44, “I^2C Controller,” describes the CPM I^2C controller.

- Chapter 45, "Parallel I/O Ports," describes the four general-purpose I/O ports of the CPM.
- Appendix A, "MPC8541E," describes the features of the MPC8541E, how it differs from the MPC8555E, and provides additional material and specifics for using this reference manual for the MPC8541E.
- Appendix B, "Revision History," lists the major differences between revisions of the *MPC8555E PowerQUICC™ III Integrated Processor Reference Manual*.
- This reference manual also includes a glossary and an index.

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 Power Architecture technology 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*, Second Edition, by David A. Patterson and John L. Hennessy

Related Documentation

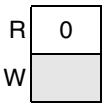
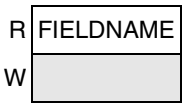
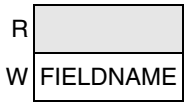
Freescale documentation is available from the sources listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering:

- *EREF: A Reference for Freescale Book E and the e500 Core*—This book provides a higher-level view of the programming model as it is defined by the Freescale embedded category implementation standards and the e500 microprocessor.
- 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.
- 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 http://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
italics	Italics indicate variable command parameters, for example, **bcctr***x* Book titles in text are set in italics Internal signals are set in lowercase italics, for example, $\overline{core\_int}$
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, **r**B	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.
x	An italicized *x* indicates an alphanumeric variable
n	An italicized *n* indicates a numeric variable
¬	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Concatenation, for example TCR[WP]||TCR[WPEXT]
	Indicates a reserved bit field in an e500 register. Although these bits can be written to as ones or zeros, they are always read as zeros.
	Indicates a reserved bit field in a memory-mapped register. Although these bits can be written to as ones or zeros, they are always read as zeros.
	Indicates a read-only bit field in a memory-mapped register.
	Indicates a write-only bit field in a memory-mapped 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.
lowercase_italics	Lowercase italics is used to indicate internal signals.
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration. For more information, see Section 3.2, "Configuration Signals Sampled at Reset."

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
CCB	Core complex 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
CPM	Communication processor module
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
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
ECM	e500 coherency module

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
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^2C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
IrDA	Infrared Data Association
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
OCeaN	On-chip network
OSI	Open systems interconnection
PCI	Peripheral component interconnect
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
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCC	Serial communication controller
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

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
TAP	Test access port
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
TSEC	Three-speed Ethernet controller
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

This part describes the many features of the MPC8555E integrated processor at an overview level. The following chapters are included:

- Chapter 1, “Overview”
- Chapter 2, “Memory Map”
- Chapter 3, “Signal Descriptions”
- Chapter 4, “Reset, Clocking, and Initialization”

Chapter 1, “Overview,” provides a high-level description of features and functionality of the MPC8555E integrated processor. It describes the MPC8555E, its interfaces, and programming model. The functional operation of the MPC8555E with emphasis on peripheral functions is also described.

Chapter 2, “Memory Map,” describes the MPC8555E memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory mapped registers with cross references to the sections detailing descriptions of 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 reset configuration signals (and the modes they define).

Chapter 4, “Reset, Clocking, and Initialization,” describes the hard and soft resets, power-on reset sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8555E.

Chapter 1
Overview

Freescale Semiconductor's MPC8555E PowerQUICC™ III integrated communications processor includes a wide range of advanced Freescale technologies, modular cores, and peripherals. Leveraging Freescale's system-on-chip (SoC) PowerQUICC III platform architecture, the MPC8555E combines the powerful e500 core and communications peripheral technology to balance processor performance with I/O system throughput. The processor is designed to offer clock speeds scaling from 533 MHz to 1 GHz.

This chapter provides a high-level description of features and functionality of the MPC8555E and MPC8541E integrated communications processors. It is written from the perspective of the MPC8555E, which is the superset device. For specifics on how to use this manual for the MPC8541E, see Appendix A, "MPC8541E."

1.1 Introduction

Freescale's MPC8555E device integrates two processing blocks: a high-performance e500 core that implements the Power Architecture™ definition of the embedded category instruction set architecture and a RISC-based communications processor module (CPM) that supports a wide range of communications peripherals. This innovative architecture is designed to reduce power consumption and offer a more balanced approach to processing than traditional processor architectures. The CPM offloads low-level peripheral communications tasks, enabling the embedded e500 core to manage high-level processing tasks. The MPC8555E device's high level of integration helps simplify board design and enhances system-level bandwidth and performance. In addition to the e500 core and CPM, the MPC8555E features an integrated security engine, a double data rate SDRAM (DDR SDRAM) memory controller, dual Gigabit Ethernet controllers, a four-channel DMA controller, dual asynchronous receiver/transmitters (DUART), and a 64-bit PCI controller that can also serve as two 32-bit PCI ports. Dual on-chip PCI support provides a cost-effective alternative to separate, discrete PCI bridges and chipsets for I/O-intensive applications that require multiple PCI interfaces. In addition to these features, the MPC8555E provides a local bus controller and I^2C support.

The MPC8555E processor features a security engine that supports DES, 3DES, MD-5, SHA-1, AES, and ARC-4 encryption algorithms, as well as offering a public key accelerator and on-chip random number generator. This embedded security core is derived from Freescale's security coprocessor product line and offers the same direct-memory access (DMA) and parallel processing capabilities, as well as the ability to perform single-pass encryption and authentication as required by widely used security protocols, such as IPsec and IEEE 802.11i® standard. Integrated security makes the MPC8555E an optimal communications processor solution for applications that require security features in concert with high performance and low system-level cost.

1.2 MPC8555E Overview

The following section provides a high-level overview of the MPC8555E features.

Figure 1-1 shows the major functional units within the MPC8555E.

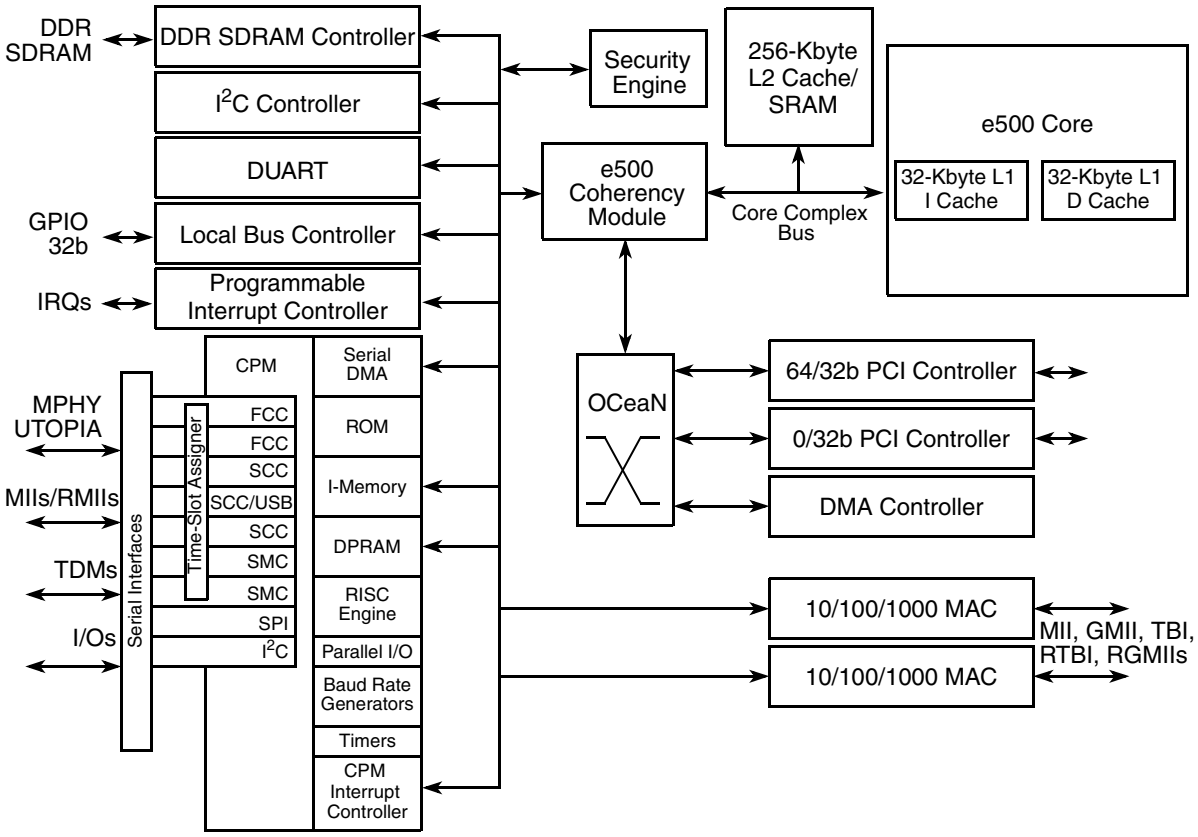


Figure 1-1. MPC8555E Block Diagram

1.2.1 Key Features

The following lists an overview of the MPC8555E feature set:

- Embedded e500 core
  - High-performance, 32-bit core that implements the embedded category of the Power Architecture technology
  - Dual-issue superscalar, 7-stage pipeline design
  - 32-Kbyte L1 instruction cache and 32-Kbyte L1 data cache with parity protection
  - Lockable L1 caches—entire cache or on a per-line basis
  - Separate locking for instructions and data
  - Single-precision floating-point operations
  - Memory management unit especially designed for embedded applications
  - Enhanced hardware and software debug support

— Dynamic power management
— Performance monitor facility

- The security engine is optimized to handle all the algorithms associated with IPSec, SSL/TLS, SRTP, 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
    – 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 Rinjdael 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- 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
- High-performance RISC CPM operating at up to 333 MHz
  — CPM software compatibility with previous PowerQUICC families
  — One instruction per clock
  — Executes code from internal ROM or instruction RAM
  — 32-bit RISC architecture
  — Tuned for communication environments: instruction set supports CRC computation and bit manipulation
  — Internal timer

— Interfaces with the embedded e500 core processor through a 32-Kbyte dual-port RAM and virtual DMA channels for each peripheral controller
— Handles serial protocols and virtual DMA
— Two full-duplex fast communications controllers (FCCs) that support the following protocols:
  – ATM protocol through two UTOPIA level II interfaces
  – IEEE 802.3® standard/Fast Ethernet
  – HDLC
  – Totally transparent operation
— Three full-duplex serial communications controllers (SCCs) support the following protocols:
  – High level/synchronous data link control (HDLC/SDLC)
  – LocalTalk (HDLC-based local area network protocol)
  – Universal asynchronous receiver transmitter (UART)
  – Synchronous UART (1x clock mode)
  – Binary synchronous communication (BISYNC)
  – Totally transparent operation
  – QMC support, providing 64 channels per SCC using only 1 physical TDM interface
— Universal serial bus (USB) controller that is full-/low-speed compliant (multiplexed on an SCC)
  – USB host mode
  – Supports USB slave mode
— Serial peripheral interface (SPI) support for master or slave
— I^2C bus controller
— Two serial management controllers (SMCs) supporting:
  – UART
  – Transparent
  – General-circuit interfaces (GCI)
— Time-slot assigner supports multiplexing of data from any of the SCCs and FCCs onto eight time-division multiplexed (TDM) interfaces. The time-slot assigner supports the following TDM formats:
  – T1/CEPT lines
  – T3/E3
  – Pulse code modulation (PCM) highway interface
  – ISDN primary rate
  – Freescale interchip digital link (IDL)
  – General circuit interface (GCI)
— User-defined interfaces
— Eight independent baud rate generators (BRGs)
— Four general-purpose 16-bit timers or two 32-bit timers

— General-purpose parallel ports—16 parallel I/O lines with interrupt capability
— Supports inverse muxing of ATM cells (IMA)

- 256 Kbytes of on-chip memory
  — Can act as a 256-Kbyte level-2 cache
  — Can act as a 256-Kbyte or two 128-Kbyte memory-mapped SRAM arrays
  — Can be partitioned into 128-Kbyte L2 cache plus 128-Kbyte SRAM
  — Full ECC support on 64-bit boundary in both cache and SRAM modes
  — SRAM operation supports relocation and is byte-accessible
  — Cache mode supports instruction caching, data caching, or both
  — External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types (stashing)
  — Eight way set-associative cache organization (1024 sets of 32-byte cache lines)
  — Supports locking the entire cache or selected lines
    – Individual line locks set and cleared through instructions or by externally mastered transactions
  — Global locking and flash clearing done through writes to L2 configuration registers
  — Instruction and data locks can be Flash cleared separately
  — Read and write buffering for internal bus accesses
- Address translation and mapping unit (ATMU)
  — Eight local access windows define mapping within local 32-bit address space
  — Inbound and outbound ATMUs map to larger external address spaces
    – Three inbound windows plus a configuration window on PCI
    – Four outbound windows plus default translation for PCI
- DDR memory controller
  — Programmable timing supporting first generation DDR SDRAM
  — 64-bit data interface, up to 333-MHz data rate
  — Four banks of memory supported, each up to 1 Gbyte
  — DRAM chip configurations from 64 Mbits to 1 Gbit with x8/x16 data ports
  — Full ECC support
  — Page mode support (up to 16 simultaneous open pages)
  — Contiguous or discontiguous memory mapping
  — Sleep mode support for self refresh DDR SDRAM
  — Supports auto refreshing
  — On-the-fly power management using CKE signal
  — Registered DIMM support
  — Fast memory access through JTAG port
  — 2.5-V SSTL2 compatible I/O

- Programmable interrupt controller (PIC)
  - Programming model is compliant with the OpenPIC architecture
  - Supports 16 programmable interrupt and processor task priority levels
  - Supports 12 discrete external interrupts
  - Supports 4 message interrupts with 32-bit messages
  - Supports connection of an external interrupt controller such as the 8259 programmable interrupt controller
  - Four global high resolution timers/counters that can generate interrupts
  - Supports additional internal interrupt sources
  - Supports fully nested interrupt delivery
  - Interrupts can be routed to external pin for external processing
  - Interrupts can be routed to the e500 core's standard or critical interrupt inputs
  - Interrupt summary registers allow fast identification of interrupt source
- Two I^2C controllers (one is contained within the CPM, the other is a stand-alone controller which is not part of the CPM)
  - Two-wire interface
  - Multiple master support
  - Master or slave I^2C mode support
  - On-chip digital filtering rejects spikes on the bus
- Boot sequencer
  - Optionally loads configuration data from serial ROM at reset through the stand-alone I^2C interface
  - Can be used to initialize configuration registers and/or memory
  - Supports extended I^2C addressing mode
  - Data integrity checked with preamble signature and CRC
- DUART
  - Two 4-wire interfaces (RXD, TXD, RTS, CTS)
  - Programming model compatible with the original 16450 UART and the PC16550D
- Local bus controller (LBC)
  - Multiplexed 32-bit address and data operating at up to 166 MHz
  - Eight chip selects support eight external slaves
  - Up to eight-beat burst transfers
  - The 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)

- Two three-speed (10/100/1000) Ethernet controllers (TSECs)
  - — Dual IEEE 802.3, 802.3u®, 802.3x®, 802.3z®, 802.3ac® standard compliant controllers
  - — Support for different Ethernet physical interfaces:
    - – 10/100/1000 Mbps IEEE 802.3 standard GMII
    - – 10/100 Mbps IEEE 802.3 standard MII
    - – 10 Mbps IEEE 802.3 standard MII
    - – 1000 Mbps IEEE 802.3z standard TBI
    - – 10/100/1000 Mbps RGMII/RTBI
  - — Full- and half-duplex support
  - — Buffer descriptors are backwards compatible with MPC8260 and MPC860T 10/100 programming models
  - — 9.6-Kbyte jumbo frame support
  - — RMON statistics support
  - — 2-Kbyte internal transmit and receive FIFOs
  - — MII management interface for control and status
  - — Programmable CRC generation and checking
- OCeaN switch fabric
  - — Three-port crossbar packet switch
  - — Reorders packets from a source based on priorities
  - — Reorders packets to bypass blocked packets
  - — Implements starvation avoidance algorithms
  - — Supports packets with payloads of up to 256 bytes
- Integrated DMA controller
  - — Four-channel controller
  - — All channels accessible by both local and remote masters
  - — Extended DMA functions (advanced chaining and striding capability)
  - — Support for scatter and gather transfers
  - — Misaligned transfer capability
  - — Interrupt on completed segment, link, list, and error
  - — Supports transfers to or from any local memory or I/O port
  - — Selectable hardware-enforced coherency (snoop/no-snoop)
  - — Ability to start and flow control each DMA channel from external 3-pin interface
  - — Ability to launch DMA from single write transaction
- PCI controllers
  - — PCI 2.2 compatible

  - — One 64-bit or two 32-bit PCI ports supported at 16 to 66 MHz
  - — Host and agent mode support, 64-bit PCI port can be host or agent, if two 32-bit ports, only one can be an agent
  - — 64-bit dual address cycle (DAC) support
  - — Supports PCI-to-memory and memory-to-PCI streaming
  - — Memory prefetching of PCI read accesses
  - — Supports posting of processor-to-PCI and PCI-to-memory writes
  - — PCI 3.3-V compatible
  - — Selectable hardware-enforced coherency
  - — Selectable clock source (SYSCLK or independent PCI_CLK)
- Power management
  - — Fully static 1.2-V CMOS design with 3.3- and 2.5-V I/O
  - — Supports power save modes: doze, nap, and sleep
  - — Employs dynamic power management
- System performance monitor
  - — Supports eight 32-bit counters that count the occurrence of selected events
  - — Ability to count up to 512 counter-specific events
  - — Supports 64 reference events that can be counted on any of the 8 counters
  - — Supports duration and quantity threshold counting
  - — Burstiness feature that permits counting of burst events with a programmable time between bursts
  - — Triggering and chaining capability
  - — Ability to generate an interrupt on overflow
- System access port
  - — Uses JTAG interface and a TAP controller to access entire system memory map
  - — Supports 32-bit accesses to configuration registers
  - — Supports cache-line burst accesses to main memory
  - — Supports large block (4-Kbyte) uploads and downloads
  - — Supports continuous bit streaming of entire block for fast upload and download
- IEEE 1149.1™ compliant, JTAG boundary scan
- 783-pin FC-PBGA package

1.3 MPC8555E Architecture Overview

The following sections describe the major functional units of the MPC8555E.

1.3.1 e500 Core Overview

The MPC8555E uses the e500 microprocessor core complex. Both the e500 core and the CPM have internal PLLs that allow independent optimization of their operating frequencies. The core and CPM frequencies are derived from either the primary PCI clock input or an external oscillator. For more information regarding the e500 core refer to the following documents:

- *EREF: A Reference for Freescale Semiconductor Book E and the e500 Core*
- *PowerPC™ e500 Core Family Reference Manual*
- *PowerPC™ e500 Application Binary Interface User's Guide*

The following is a brief list of some of the key features of the e500 core complex:

- 32-bit architecture
- Implements additional instructions, registers, and interrupts (formerly referred to as APUs). The signal processing engine (SPE) provides an extensive instruction set for 64-bit vector integer, single-precision floating-point, and fractional operations. The SPFP provides scalar (32-bit) single-precision floating-point instructions.

NOTE

The SPE and SPFP functionality will be implemented in all PowerQUICC III devices. However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or SPFP instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale offers a libcfsl_e500 library that uses SPE and SPFP instructions. Freescale will also provide future libraries to support next generation PowerQUICC devices.

- L1 cache structure
  - 32–Kbyte, 32-byte line, eight way set-associative instruction cache
  - 32–Kbyte, 32-byte line, eight way set-associative data cache
  - 1.5-cycle cache array access, 3-cycle load-to-use latency
  - Pseudo-LRU replacement algorithm
  - Copy-back data cache
- Dual-dispatch superscalar
- Precise exception handling
- Seven-stage pipeline control
- Instruction unit

  - Twelve-entry instruction queue
  - Full hardware detection of interlocks
  - Dispatch up to two instructions per cycle
  - Dispatch serialization control
  - Register dependency resolution and renaming
- Branch unit (BU)
  - Dynamic branch prediction
  - Two-entry branch instruction queue (BIQ)
  - Executes all branch and CR logical instruction
- Completion unit
  - As many as 14 instructions allowed in 14-entry completion queue
  - In-order retirement of up to two instructions per cycle
  - Completion and refetch serialization control
  - Synchronization for all instruction flow changes—interrupts and mispredicted branches
- Two simple execution units that perform:
  - Single-cycle add and subtract
  - Single-cycle shift and rotate
  - Single-cycle logical operations
- Multiple-cycle execution unit (MU)
  - Four-cycle latency for multiplication (including floating-point multiply instructions).
  - Variable-latency divide: 4, 11, 19, and 35 cycles for all divide instructions. Note that the MU allows divide instructions to bypass the second two MU pipeline stages, freeing those stages for other MU instructions to execute in parallel.
  - Four-cycle floating-point multiply
  - Four-cycle floating-point add and subtract
  - Single-precision floating-point operations
- Load/store unit (LSU)
  - Three-cycle load latency
  - Fully pipelined
  - Four-entry load queue allows up to four load misses before stalling
  - Can continue servicing load hits when load queue is full
  - Six-entry store queue allows full pipelining of stores
- Cache coherency
  - Bus support for hardware-enforced coherency (bus snooping)
  - Core complex bus (CCB)
  - High-speed, on-chip local bus with data tagging
  - 32-bit address bus
  - 60x-like address protocol with address pipelining and retry/copyback

— Two general-purpose read data, one write data bus
— 128-bit data plus parity/tags (each data bus)
— Supports out-of-order reads, in-order writes
— Little to no data bus arbitration logic required for native systems
— Easily adaptable to 60x-like environments
— Supports one-level pipelining of addresses with address-retry responses

- Extended exception handling
  — Supports embedded category interrupt model
    – Interrupt vector prefix register (IVPR)
    – Vector offset registers (IVORs) 0–15 and 32–35
    – Exception syndrome register (ESR)
    – Preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rfci** instruction
    – A separate set of resources for machine-check interrupts
    – SPE unavailable exception
    – Floating-point data exception
    – Floating-point round exception
    – Performance monitor
- Memory management unit (MMU)
  — Data L1 MMU
    – Four-entry, fully-associative TLB array for variable-sized pages
    – 64-entry, four way set-associative TLB for 4-Kbyte pages
  — Instruction L1 MMU
    – Four-entry, fully-associative TLB array for variable-sized pages
    – 64-entry, four way set-associative TLB for 4-Kbyte pages
  — Unified L2 MMU
    – 16-entry, fully-associative TLB array for variable-sized pages
    – 256-entry, two way set-associative TLB for 4-Kbyte pages
  — Software reload for TLBs
  — Virtual memory support for as much as 4 Gbytes (2^{32}) of virtual memory
  — Real memory support for as much as 4 Gbytes (2^{32}) of physical memory
  — Support for big-endian and true little-endian memory on a per-page basis
- Power management
  — Low power design
  — 1.2-V design
  — Dynamic power management on the core minimizes power consumption of functional unit, such as execution units, caches, and MMUs, when they are idle.
  — Power-saving modes: standby and shutdown

— Internal clock multipliers of 2x, 2.5x, and 3x from bus clock

- Testability
  - — LSSD scan design
  - — JTAG interface
  - — ESP support
- Reliability and serviceability
  - — Internal code parity
  - — Parity checking on e500 local bus

1.3.2 Integrated Security Engine (SEC)

A block diagram of the integrated security engine's internal architecture is shown in Figure 1-2. The bus interface module is designed to transfer 64-bit words between the internal bus and any register inside the SEC.

An operation begins with a write of a pointer to a crypto-channel fetch register which 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.

The SEC functionality is compatible with code written for the Freescale MPC185 encryption device.

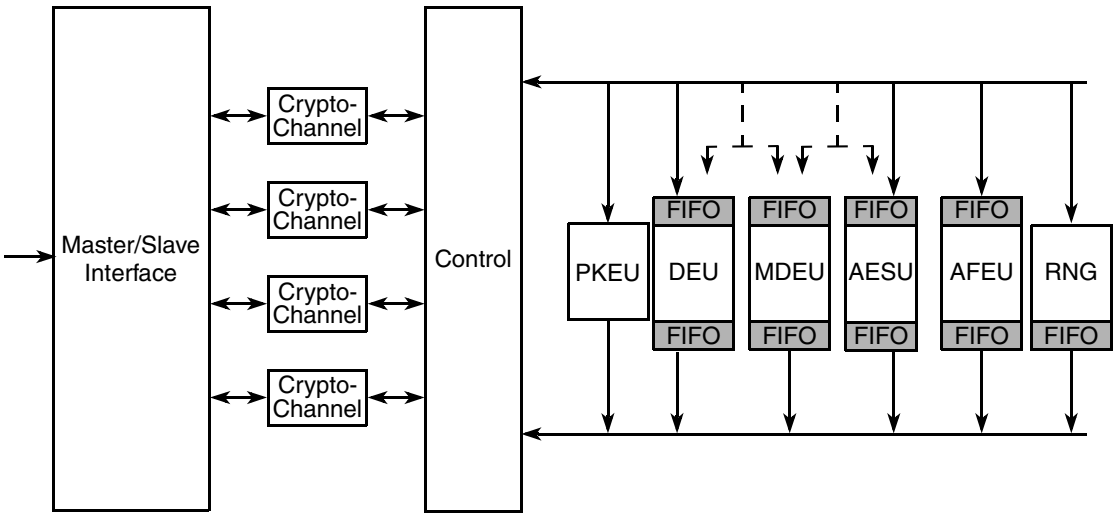


Figure 1-2. Integrated Security Engine Functional Blocks

1.3.3 Communications Processor Module (CPM)

The CPM contains features that allow the MPC8555E to excel in a variety of applications targeted for networking and telecommunication markets. The MPC8555E CPM is a superset of the MPC8260 PowerQUICC II, with enhanced communications processor (CP) performance. The CPM also has additional hardware and microcode routines that support high bit rate protocols like ATM (up to 155 Mbps full-duplex) and Fast Ethernet (100 Mbps full-duplex). The MPC8555E CPM features are very similar to those of the PowerQUICC II devices, such as the MPC8272. Existing PowerQUICC II code should port easily to the MPC8555E.

Figure 1-3 shows the major functional units within the MPC8555E CPM.

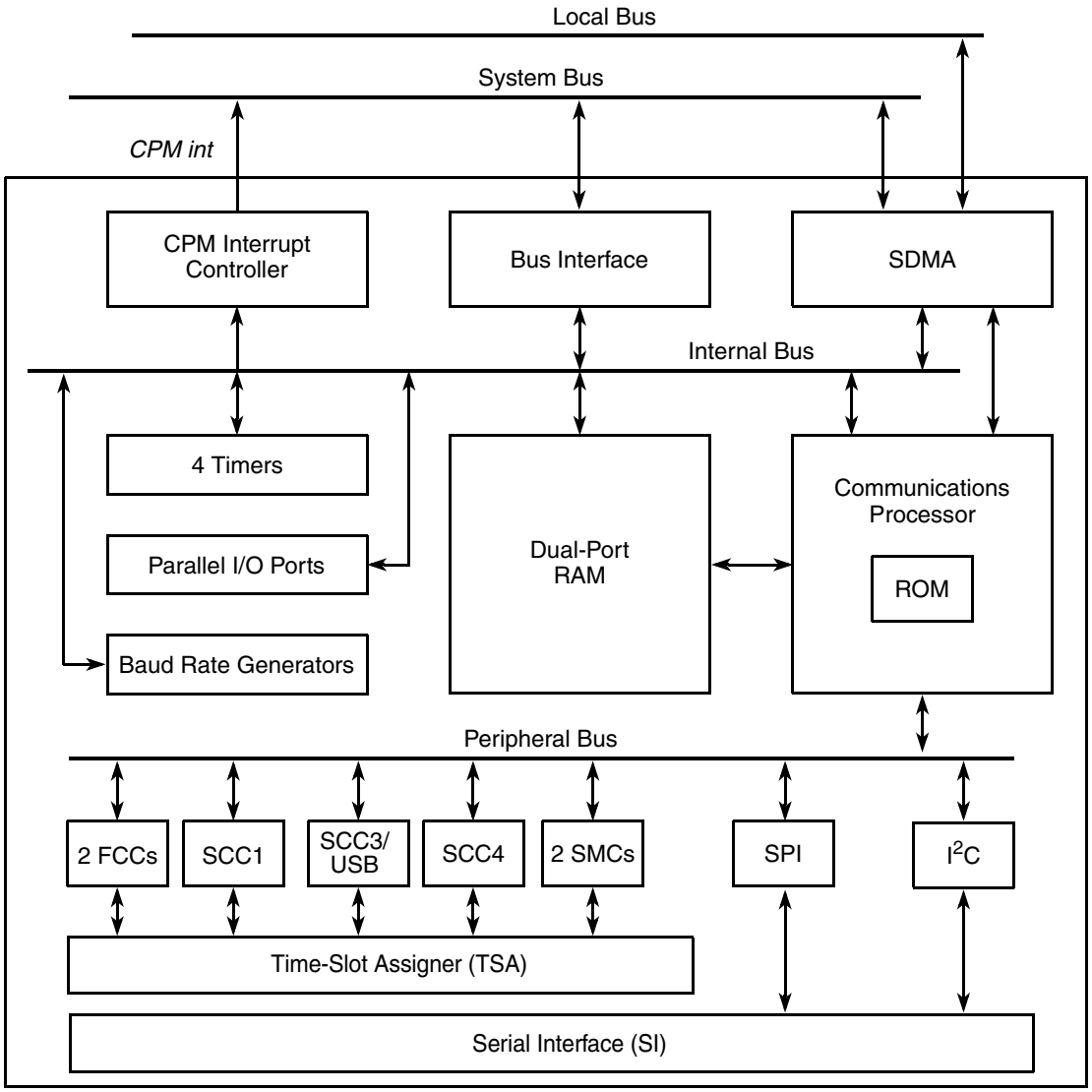


Figure 1-3. MPC8555E Communications Processor Module (CPM) Block Diagram

The following list summarizes the major features of the CPM:

- The communications processor (CP) is an embedded 32-bit RISC controller residing on a separate bus (CPM local bus). With this separate bus, the CP does not affect the performance of the e500 core. The CP handles the lower-layer tasks and DMA control activities, leaving the e500 core free to handle higher-layer activities. The CP has an instruction set optimized for communications but that can also be used for general-purpose applications, relieving the system core of small, often repeated tasks.
- Two serial DMAs (SDMAs), one associated with the local bus and one associated with the e500 coherency module (ECM), handling transfers simultaneously.
- Two full-duplex, serial fast communications controllers (FCCs) supporting ATM (155 Mbps) protocol through two UTOPIA level II interfaces. Or the FCCs may be configured to support IEEE 802.3 Fast Ethernet, or HDLC up to E3 rates (45 Mbps). Each FCC can be configured to transmit fully transparent and receive HDLC data or vice-versa.
- Three full-duplex serial communications controllers (SCCs) supporting, high-level synchronous data link control (HDLC), local talk, UART, synchronous UART, BISYNC, or transparent. They also support QUICC multichannel control (QMC), which can run 64 channels per SCC using only one physical TDM interface. SCC3 can be configured as a USB controller which is full/low speed compliant.
- One full-duplex serial peripheral interface (SPI) providing a synchronous, character-oriented channel that supports a four-wire interface for communication with other microprocessors, peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.
- One I^2C bus controller providing communication with other I^2C capable devices, see Section 1.3.8, "I^2C Controllers." There are actually two I^2C controllers on the MPC8555E, this one in the CPM and a separate standalone I^2C controller.
- Two full-duplex serial management controllers (SMCs) that can be configured as UART, transparent, or general-circuit interfaces (GCI).
- Time-slot assigner (TSA) that supports multiplexing of data from any of the FCCs, SCCs, or SMCs.

1.3.4 On-Chip Memory Unit

The MPC8555E contains an internal 256-Kbyte memory array that can be configured as memory-mapped SRAM or as a look-aside L2 cache. The array can also be divided into two 128-Kbyte arrays, one of which may be used as cache and the other as SRAM.

The memory controller for this array connects to the core complex bus (CCB) and communicates through 128-bit read and write buses to the e500 core and the MPC8555E system logic.

The on-chip memory unit contains:

- 256-Kbytes of on-chip memory
  - — L2 cache partitioning is configurable
    - – Can act as a 256-Kbyte L2 cache
      - – 256-Kbyte array organized as 1024 eight way sets of 32-byte cache lines

  - Array can be partitioned into 128-Kbyte L2 cache and 128-Kbyte memory mapped SRAM
  - Can act as two 128-Kbyte memory-mapped SRAM arrays or a 256-Kbyte SRAM region
- SRAM operation is byte-accessible
- Data ECC on 64-bit boundaries (single-error correction, double-error detection)
- Tag parity (1 bit covering all tag bits)
- Cache mode supports instruction caching, data caching, or both
- External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types
- Separate locking for instructions and data so that locks can be set and cleared separately
- Supports locking the entire cache or selected lines
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges
- Flash clearing done through writes to L2 configuration registers
- Locks for the entire cache may be set and cleared by accesses to memory-mapped control registers

1.3.4.1 On-Chip Memory as Memory-Mapped SRAM

When the on-chip memory is configured as an SRAM, the 256 Kbytes of memory can be configured to reside at any aligned location in the memory map. It is byte-accessible and fully ECC protected using read-modify-write transactions for sub-cacheline transactions. I/O devices can access the SRAM by marking transactions global so that they are directed to the core complex bus (CCB).

1.3.4.2 On-Chip Memory as L2 Cache

The MPC8555E on-chip memory arrays include a 256-Kbyte data array, an address tag array, and a status array.

The data array is organized as 1024 sets of 8 cache lines. Each cache line size is 32 bytes. The replacement policy within each eight way set is governed by a pseudo-LRU algorithm. The data is protected with ECC and the tag array is protected by parity.

The L2 cache tags are non-blocking for efficient load/store and snooping operations. The L2 cache can be accessed internally while a load miss is pending (allowing hits under misses). Subsequent to a load miss updating the memory, loads or stores can occur to that line on the very next cycle.

The L2 status array maintains status bits for each line that are used to determine the status of the line. Different combinations of these bits result in different L2 states. Note that because the cache is always write-through, there is no modified state. The status bits include:

- V—Valid
- IL—Instruction locked
- DL—Data locked

All accesses to the L2 memory are fully pipelined so back-to-back loads and stores can have single-cycle throughput.

The cache can be configured to allocate instructions-only, data-only, or both. It can also be configured to allocate global I/O writes that correspond to a programmable address window or that use a special transaction type (stashing). In this way, DMA engines or I/O devices can force data into the cache.

Line locks can be set in a variety of ways. The architecture defines instructions that explicitly set and clear locks in the L2. These instructions are supported by the core complex and the L2 controller. In addition, the L2 controller can be configured to lock all lines that fall into either of two specified address ranges when the line is allocated. Finally, the entire cache can be locked by writing to a configuration register in the L2 cache controller.

The status array tracks line locks as either instruction locks or data locks for each line, and the status array supports flash clearing of all instruction locks or data locks separately by writes to configuration registers in the L2 controller.

1.3.5 e500 Coherency Module (ECM)

The e500 coherency module (ECM) provides a mechanism for I/O-initiated transactions to snoop the bus between the e500 core and the integrated L2 cache in order to maintain coherency across local cacheable memory. It also provides a flexible switch-type structure for core and I/O-initiated transactions to be routed or dispatched to target modules on the device.

1.3.6 DDR SDRAM Controller

The MPC8555E supports DDR-I SDRAM that operates at up to 166 MHz (333-MHz data rate). The memory interface controls main memory accesses and provides for a maximum of 3.5 Gbytes of main memory. The memory controller can be configured to support the various memory sizes through software initialization of on-chip configuration registers.

The MPC8555E supports a variety of SDRAM configurations. SDRAM banks can be built using DIMMs or directly-attached memory devices. Fifteen multiplexed address signals provide for device densities of 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, and 1 Gbit. Four chip select signals support up to four banks of memory. The MPC8555E supports bank sizes from 64 Mbytes to 1 Gbyte. Nine column address strobes (MDM[0:8]) are used to provide byte selection for memory bank writes.

The MPC8555E can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 16 simultaneously open pages can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save 3 to 4 clock cycles from subsequent burst accesses that hit in an active page.

The MPC8555E supports error checking and correction (ECC) for system memory. Using ECC, the MPC8555E detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

The MPC8555E can invoke a level of system power management by asserting the MCKE SDRAM signal on-the-fly to put the memory into a low-power sleep mode.

1.3.7 Programmable Interrupt Controller (PIC)

The programmable interrupt controller (PIC) implements the necessary functions to provide a flexible solution for a general purpose interrupt control. The interrupt controller unit implements the logic and programming structures of the OpenPIC architecture. The MPC8555E interrupt controller unit supports its processor core and provides for 12 external interrupts (with fully nested interrupt delivery), 4 message interrupts, internal-logic driven interrupts, and 4 global high resolution timers. Up to 16 programmable interrupt priority levels are supported.

The interrupt controller unit can be bypassed to allow use of an external interrupt controller. Inter-processor interrupt (IPI) communication is supported through the external interrupt and core reset signals of different processor cores on the same device. The four IPIs are only used for self-interrupt in a single-core device such as the MPC8555E.

1.3.8 I^2C Controllers

There are two inter-IC (IIC or I^2C) controllers on the MPC8555E that are full/low speed compliant. One is contained within the CPM and multiplexed on an SCC. The second one is a stand-alone I^2C controller. Both of the I^2C controllers provide an I^2C, two-wire, bi-directional serial bus that provides a simple, efficient method of data exchange between devices. The synchronous, multi-master bus of the I^2C controllers allow the MPC8555E to exchange data with other I^2C devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus (serial data SDA and serial clock SCL) minimizes the interconnections among devices, and provides application expansion and system development.

The I^2C controller is a true multiple-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 I^2C controller consists of a transmitter/receiver unit, a clocking unit, and a control unit. The I^2C unit supports general broadcast mode, and has on-chip filtering that rejects spikes on the bus.

1.3.9 Boot Sequencer

The MPC8555E provides a boot sequencer that uses the stand alone I^2C controller interface to access an external serial ROM and loads the data into the MPC8555E configuration registers. The boot sequencer is enabled by a configuration pin that is sampled at the negation of the MPC8555E hardware reset signal. If enabled, the boot sequencer holds the MPC8555E processor core in reset until the boot sequence is complete. If the boot sequencer is not enabled, the processor core exits reset and fetches boot code in default configurations.

1.3.10 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The MPC8555E includes a DUART intended for use in maintenance, bringing-up, and debugging of systems. The MPC8555E provides a standard four-wire handshake (TXD, RXD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$) 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, line status, and MODEM status.

The MPC8555E DUART supports a full-duplex operation. It is compatible with the PC16450 and PC16550 programming models. Also, 16-byte FIFOs are supported for both the transmitter and the receiver.

Software programmable baud 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.

1.3.11 Local Bus Controller (LBC)

The MPC8555E local bus controller (LBC) allows connections with a wide variety of external memories, DSPs, and ASICs. Three separate state machines share the same external pins and can be programmed separately to access different types of devices. The general purpose chip select machine (GPCM) controls accesses to asynchronous devices using a simple handshake protocol. The user programmable machine (UPM) can be programmed to interface to synchronous devices or custom ASIC interfaces. The SDRAM controller provides access to standard SDRAM. Each chip select can be configured so that the associated chip interface can be controlled by the GPCM, UPM, or SDRAM controller. All may exist in the same system.

The GPCM provides a flexible asynchronous interface to SRAM, EPROM, FEPROM, ROM, and other devices such as asynchronous DSP host interfaces and CAMs. Minimal glue logic is required. Handshake signals can be configured to transition on fractions of the system clock. The GPCM does not support bursting.

The UPM allows an extremely flexible interface in which the programmer configures each of a set of general purpose protocol signals by writing the transition pattern into a memory array. The UPM supports synchronous and bursting interfaces. It also supports multiplexed addressing so that a simple DRAM interface can be implemented. The UPM is entirely flexible in order to provide a very high degree of customization with respect to both asynchronous and burst-synchronous interfaces, which permits glueless or almost glueless connection to burst SRAM, custom ASIC, and synchronous DSP interfaces.

The LBC provides a synchronous DRAM (SDRAM) machine that provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices. An internal DLL (delay-locked loop) for bus clock generation ensures improved data setup margins for board designs. The SDRAM machine can optimize burst transfers and exploits interleaving to maximize data transfer bandwidth and minimize access latency. Programmable row and column address multiplexing allows a variety of SDRAM configurations and sizes to be supported without hardware changes.

1.3.12 Three-Speed Ethernet Controllers (10/100/1Gb)

The MPC8555E has two on-chip three-speed Ethernet controllers (TSECs). The TSECs incorporate a media access control sublayer (MAC) that supports 10- and 100-Mbps, and 1Gbps Ethernet/802.3 networks with MII, GMII, RGMII, RTBI, and TBI physical interfaces. The TSECs include 2-Kbyte receive and transmit FIFOs, and DMA functions.

The buffer descriptors are based on the MPC8260 and MPC860T 10/100 programming models.

The MPC8555E TSECs support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 Kbytes. Frame headers and buffer descriptors can be forced into the L2 cache to speed classification or other frame processing.

1.3.13 Integrated DMA

The MPC8555E 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 four-channel DMA controller allows chaining (both extended and direct) through local memory-mapped chain descriptors. Scattering, gathering, and misaligned transfers are supported. In addition, advanced capabilities such as stride transfers and complex transaction chaining are supported.

DMA transfers can be initiated by a single write to a configuration register. There is also support for external control of transfers using $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, and $\overline{\text{DMA\_DDONE}}$ handshake signals.

Local attributes such as snoop and L2-write stashing can be specified by DMA descriptors.

Interrupts are provided on a completed segment, link, list, chain, or on an error condition. Coherency is selectable and hardware enforced (snoop/no snoop).

1.3.14 PCI Controller

The MPC8555E 32-/64-bit PCI controller is compatible with the *PCI Local Bus Specification, Revision 2.2*. The interface can function as a host or agent bridge interface. The PCI interface supports 64-bit addressing and 32- or 64-bit data buses, providing either two independent 32-bit interfaces or one 64-bit interface.

As a master, the MPC8555E supports read and write operations to the PCI memory space, the PCI I/O space, and the PCI configuration space. Also, the MPC8555E can generate PCI special-cycle and interrupt-acknowledge commands. As a target, the MPC8555E supports read and write operations to system memory as well as configuration accesses.

An internal arbiter can be used to support up to five external masters. A round robin arbitration algorithm with two priority levels is used.

1.3.15 Power Management

In addition to low-voltage operation and dynamic power management in its execution units, the MPC8555E supports four power consumption modes: full-on, doze, nap, and sleep. The three low-power modes: doze, nap, and sleep, can be entered under software control in the e500 core or by external masters accessing a configuration register.

Doze mode suspends execution of instructions in the e500 core. The core is left in a standby mode in which cache snooping and time base interrupts are still enabled. Device logic external to the processor core is fully functional in this mode.

Nap mode shuts down clocks to all the e500 functional units except the time base, which can be disabled separately. No snooping is performed in nap mode, but the device logic external to the processor core is fully functional.

Sleep mode shuts down not only the e500 core, but also all of the MPC8555E I/O interfaces as well. Only the interrupt controller and power management logic remain enabled so that the device can be awakened.

1.3.16 Clocking

The MPC8555E takes in the SYSCLK signal as an input to the device PLL, and multiplies it by an integer from 1 to 16 to generate the internal platform clock. This platform clock is the same frequency as the DDR DRAM data rate (266 or 333 MHz). The core complex bus (CCB) and L2 cache also run at this frequency. The e500 core uses the platform clock as an input to its PLL, which multiplies it again by 2, 2.5, 3, or 3.5 to generate the core clock.

Six differential clock pairs are generated for DDR DRAMs. DLLs are used in the local bus memory controller (LBC) to generate two clock outputs.

The PCI interface can be clocked by SYSCLK or optionally by an independent PCI_CLK. There is no frequency or phase relationship required between SYSCLK and PCI_CLK.

1.3.17 Address Map

The MPC8555E supports a flexible physical address map. Conceptually, the address map consists of local space and external address space. The local address map is 4 Gbytes. The MPC8555E can be made part of a larger system address space through the mapping of translation windows. This functionality is included in the address translation and mapping units (ATMUs). Both inbound and outbound translation windows are provided. The ATMUs allows the MPC8555E to be part of larger address maps such as the PCI 64-bit address environment.

1.3.18 OCeaN Switch Fabric

In order to reduce the strain on the core interconnects with the addition of new functional blocks in this generation of the PowerQUICC family, an on-chip non-blocking crossbar switch fabric called OCeaN (on-chip network) has been integrated to decrease contention, decrease latency, and increase bandwidth. This revolutionary non-blocking crossbar fabric allows for full-duplex port connections at 128 Gbps concurrent throughput, and independent per-port transaction queuing and flow control.

1.4 Data Processing Overview

Protocol data units (PDUs) can navigate through the various MPC8555E I/O ports in three ways. In the first, data is processed by the MPC8555E CPM (as it is received and transmitted through the UTOPIAs, MIIs, and TDMs associated with the CPM) and the local bus. In the second method, data is received by any of the available I/O ports, is sent through the on-chip switch fabric, and is finally transmitted on the target I/O port without the use of the ECM. Lastly, data can be routed from any I/O port to any other I/O port through the ECM.

1.4.1 Processing Between the CPM and Local Bus

In this case, the MPC8555E stores data in buffers that reside in SDRAM on the local bus. These buffers are each referenced by a buffer descriptor (BD), which may reside in one of two tables (Rx-receive and Tx-transmit) typically placed in DPRAM. The following is a general overview of how incoming data is processed by the CPM (refer to Figure 1-4).

1. When Rx data arrives on the I/O port, it is decoded; the PDU is delineated from the incoming data stream.
2. Next, data is converted from its serial form into a parallel form and then loaded into the Rx FIFO.
3. When the Rx FIFO is filled to a set threshold, the respective communication channel signals the CPM for service.
4. The CPM accesses the next available RxBD in the RxBD table (pointed to by a register in the channel's parameter RAM table). The BD defines the main memory location where the data is to be placed, as well as the length of this buffer. Data is then moved from the Rx FIFO to a temporary storage location by the CPM.
5. Data is finally moved from this temporary storage location to main memory through DMA transactions. The status and control bits of the BD are updated, and the BD is closed.
6. A CPU interrupt is instantiated to notify the core that a new packet has been received.

Because FIFOs are typically smaller than the incoming PDU, steps 1–3 may be repeated several times to store the entire packet. There may also be times when the incoming PDU is larger than the buffer length defined in the RxBD. In such cases, steps 4–5 may be repeated and several BDs may be opened and closed to store the entire PDU.

When the transmit portion of the communication channel is enabled, the CPM starts with the first BD in the TxBD table (pointed to by a register in the channel's parameter RAM table), polling the ready R bit of the BD to verify that the buffer is ready for transmission.

7. When the BD is marked ready, the data is moved from the main memory buffer to a temporary storage location through DMA transactions.
8. The CPM moves data from the temporary memory location to the Tx FIFO.
9. Data is taken from the Tx FIFO in its parallel form and serialized.
10. When the Tx FIFO is emptied to a set threshold, the communication channel signals the CPM for more data in order to maintain throughput.
11. The serialized data is lastly encoded and transmitted on the I/O port.

Again, because the buffer pointed to by the Tx BD is typically larger than the Tx FIFO, the communication channel may make several iterations of steps 7–10 to load and transmit the entire PDU.

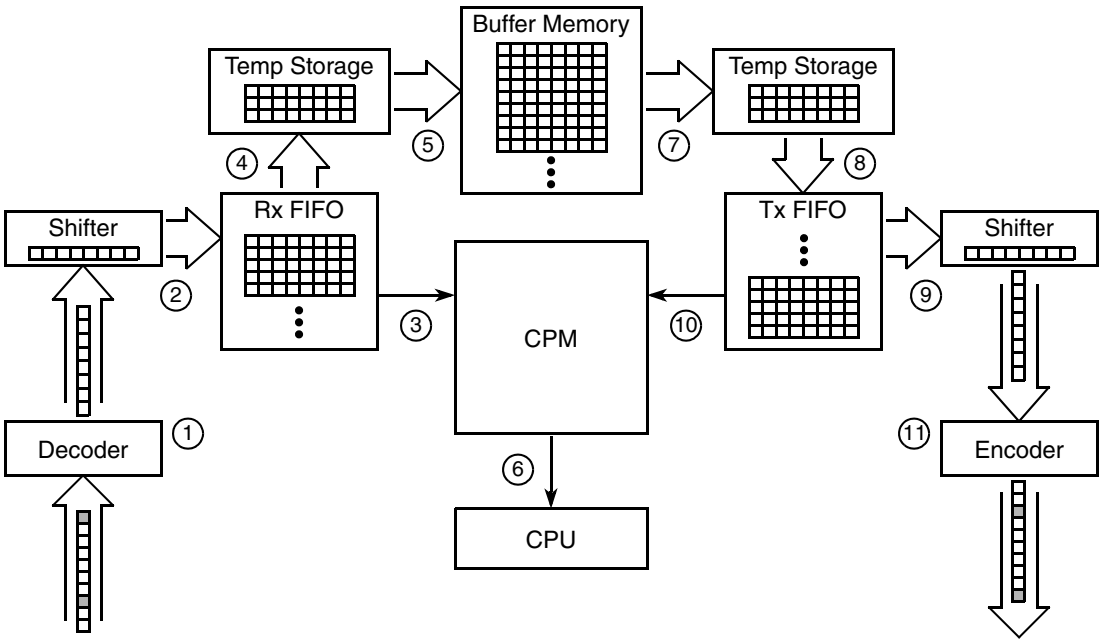


Figure 1-4. Data Processing Within the CPM

1.4.2 Processing Across the On-Chip Fabric

When processing across the on-chip fabric, the ATMUs at each fabric port are used to determine the flow of data across the MPC8555E. The ATMUs at each fabric port are responsible for generating a fabric port destination ID as well as a new local device address. The port ID and local address are based on the programmed destination of the transaction. The following is a general overview of how the ATMUs process transactions over the on-chip fabric (refer to Figure 1-5).

1. When a transaction on one of the fabric ports begins, the ATMU on the origination port translates the programmed destination address into both a destination fabric port ID and a local device address.
2. The data is then processed across the on-chip fabric from the origination port to the destination port.
3. If the destination port connects off-chip (for example, to a PCI device), the local device address is translated by the destination port ATMU to an outbound address with respect to the destination port's memory map, and the data is processed accordingly.

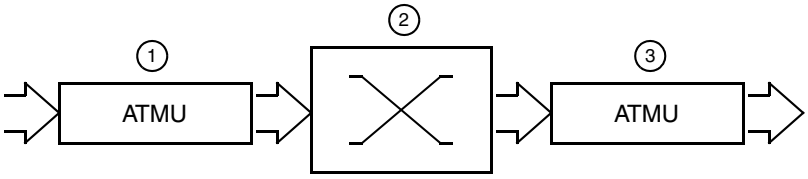


Figure 1-5. Processing Transactions Across the On-Chip Fabric

1.4.3 Data Processing with the e500 Coherency Module

Processing through the ECM is similar to processing between the CPM and local bus or across the on-chip fabric (in the sense of how data is received and transmitted) with the exception that the transaction passes through the ECM. The purpose of the ECM is to provide a means for any I/O transaction to maintain coherency with the cacheable DDR SDRAM and the local bus memory (except in the case where the CPM is directly accessing the local bus). However, simply because the ECM is used does not make transactions across it coherent. The e500 and L2 cache are snooped to maintain coherency only if the transaction across the ECM is designated as global (GBL bit set). Otherwise, the transaction passes through the ECM using the ECM as a simple conduit to get to its destination. In essence, only global transactions across the ECM are coherent transactions; all others (between the CPM and the local bus, and across the on-chip fabric) are non-coherent.

While transactions between the CPM and local bus are considered non-coherent because the CPM typically interfaces directly to the local bus (where its buffers are stored), CPM transactions can be made coherent. ATM transactions on a per-connection and direction basis can be set as coherent by setting the necessary bits in the receive and transmit connection tables. Coherency of MCC transactions per logical channel is determined by bits set in TSTATE. FCC and SCC transactions per physical channel and direction can be programmed as coherent by asserting the appropriate bits in the FCC and SCC functional code registers, respectively.

1.5 Compatibility Issues

This section describes some software and hardware compatibility issues.

1.5.1 Software

The MPC8555E CPM features are similar to those in the previous generation MPC8260. The code ports easily from previous devices to the MPC8555E, except for new protocols. During the definition of this device, an effort was made to maintain compatibility wherever possible.

Note that the MPC8555E initialization code requires changes from the MPC8260 initialization code (Freescale will provide the reference code).

1.5.2 MPC8555E Hardware

As the MPC8555E family migrates to smaller geometries, the core voltage will reduce from 1.2 V to lower voltages. A programmable voltage regulator is recommended for future compatibility. See the MPC8555E hardware specifications for the electrical requirements and the AC and DC characteristics.

1.5.3 Communications Protocol Table

Table 1-1 summarizes available protocols for each communications port.

Table 1-1. MPC8555E Protocols

Protocol	Port			
	TSEC	FCC	SCC	SMC
ATM (UTOPIA)		√		
ATM (Serial)				
1000BaseT	√			
100BaseT	√	√		
10BaseT	√	√		
HDLC		√	√	
HDLC_BUS			√	
Transparent		√	√	√
UART			√	√
Multichannel (QMC)			√	
USB			√	

1.5.4 MPC8555E Configurations

The MPC8555E offers flexibility in configuring the device for specific applications. The functions mentioned in the above sections are all available in the device, but not all of them can be used at the same time. This does not imply that the device is not fully activated in any given implementation. The CPM architecture has the advantage of using common hardware resources for many different protocols and applications. Two factors limit the functionality in any given system: pinout and performance.

1.5.5 Pin Configurations

To maximize the efficiency of device pins, some pins have multiple functions. In some cases choosing a function may preclude the use of another function.

1.5.6 Communications Performance

The CPM is designed to handle an aggregate of 1 Gbps on the communications channels running at 333 MHz. Performance depends on a number of factors:

- Channel rate versus CPM clock frequency for adequate polling of communications channels for service
- Channel rate and protocol versus CPM clock frequency for CP protocol handling
- Channel rate and protocol versus bus bandwidth
- Channel rate and protocol versus system core clock for adequate protocol handling

The second item above is addressed in this section—the CP's ability to handle high bit-rate protocols. Slow bit-rate protocols do not significantly affect those numbers.

Table 1-2 shows the peak CPM performance of various protocols under the assumption that only one of those protocols is running at a given time. The ATM numbers shown also assume that the local bus is used exclusively by the CPM and enough bandwidth on the DDR memory system is available for the CPM (implying that other resources like the PCI controllers, TSECs, DMA controller, and CPU do not all operate at their maximum performance). The frequency specified is the minimum CPM frequency necessary to run the mentioned protocols concurrently in full-duplex.

Table 1-2. Peak CPM Performance by Protocol

<table>
<tr><th colspan="3" rowspan="2">Protocol</th><th colspan="3">Frame Size</th></tr>
<tr><th>1024 Bytes</th><th>128 Bytes</th><th>64 Bytes</th></tr>
<tr><td>Ethernet</td><td colspan="2">FCC: 100Base T</td><td colspan="3">2 × 100Base T Full Duplex = 400 Mbps</td></tr>
<tr><td rowspan="5">ATM</td><td rowspan="2">FCC: AAL5</td><td>No bus limitation</td><td rowspan="2">≥1000 Mbps aggregated</td><td colspan="2" rowspan="2">≥900 Mbps aggregated</td></tr>
<tr><td>Connection tables on local bus</td></tr>
<tr><td rowspan="2">FCC: AAL0</td><td>No bus limitation</td><td colspan="3" rowspan="2">≥1000 Mbps aggregated</td></tr>
<tr><td>Connection tables on local bus</td></tr>
<tr><td colspan="2">FCC: AAL2 CPS</td><td colspan="3">33–242 Mbps depending on PDU size</td></tr>
</table>

These performance estimates assume the CPM is operating at 333 MHz, and that data is stored in SDRAM on the local bus operating at 166 MHz.

1.6 Reference Manual Revision History

A list of the major differences between revisions of the *MPC8555E PowerQUICC™ III Integrated Processor Family Reference Manual* is provided in Appendix B, "Revision History."

Chapter 2
Memory Map

This chapter describes the MPC8555E memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of each.

2.1 Local Memory Map Overview and Example

The MPC8555E provides an extremely 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. DMA engines also see this same local memory map. All memory accessed by the MPC8555E 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 eight local access windows. Each of these windows map a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. 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. The target interface is specified using the codes shown in Table 2-1.

Table 2-1. Target Interface Codes

Target Interface	Target Code
PCI 1	0000
PCI 2	0001
Local bus	0100
DDR SDRAM	1111

Figure 2-1 shows an example memory map.

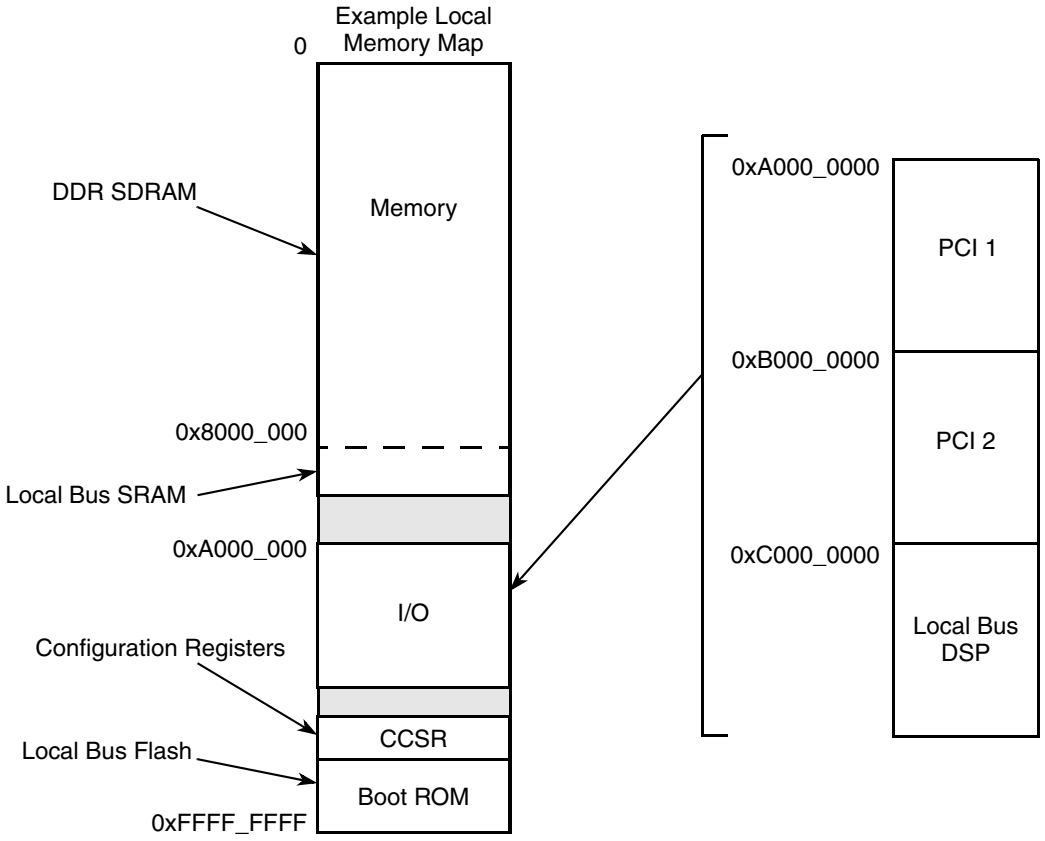


Figure 2-1. Local Memory Map Example

Table 2-2 shows one corresponding set of local access window settings.

Table 2-2. Local Access Windows Example

Window	Base Address	Size	Target Interface
0	0x0000_0000	2 Gbytes	0b1111 (DDR SDRAM)
1	0x8000_0000	1 Mbyte	0b0100 (local bus)
2	0xA000_0000	256 Mbytes	0b0000 (PCI 1)
3	0xB000_0000	256 Mbytes	0b0001 (PCI 2)
4	0xC000_0000	256 Mbytes	0b0100 (local bus)
5–7	Unused		

In this example, it is not necessary to use a local access window to specify the location of the boot ROM because it is in the default location at the highest 8 Mbytes of memory (see Section 4.4.3.3, “Boot ROM Location”). Neither is it required to define a local access window to describe the range of memory used for memory-mapped registers because this is a fixed 1-Mbyte space pointed to by CCSRBAR. See Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register (CCSRBAR).” However, note that the e500 core only provides one default TLB entry to access boot code and it allows for accesses

within the highest 4 Kbytes of memory. In order to access the full 8 Mbytes of default boot space (and the 1 Mbyte of CCSR space), additional TLB entries must be set up within the e500 core for mapping these regions.

2.2 Address Translation and Mapping

Four distinct types of translation and mapping operations are performed on transactions in the MPC8555E. These are as follows:

- Mapping a local address to a target interface
- Assigning attributes to transactions
- Translating local 32-bit addresses to external address spaces
- Translating external addresses to the local 32-bit address space

The local access windows perform target mapping for transactions within the local address space. No address translation is performed by the local access windows.

Outbound ATMU windows perform the mapping from the local 32-bit address space to the external address spaces of the PCI controllers, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.

Inbound ATMU windows perform the address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and also map the transaction to its target interface. Note that in mapping the transaction to the target interface, an inbound ATMU window performs a similar function as the local access windows. The target mappings created by an inbound ATMU must be consistent with those of the local access windows. That is, if an inbound ATMU maps a transaction to a given local address and a given target, a local access window must also map that same local address to the same target.

All of the configuration registers that define translation and mapping functions use the concept of translation or mapping windows, and all follow the same register format. Table 2-3 summarizes the general format of these window definitions.

Table 2-3. Format of ATMU Window Definitions

Register	Function
Translation address	High-order address bits defining location of the window in the target address space
Base address	High-order address bits defining location of the window in the initial address space
Window size/attributes	Window enable, window size, target interface, and transaction attributes

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

2.2.1 SRAM Windows

The on-chip memory array of the MPC8555E can be configured as a memory-mapped SRAM of 128 or 256 Kbytes. Configuration registers in the L2 cache controller set the base addresses and sizes for these windows. When enabled, these windows supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR. It is possible to have SRAM windows overlap local access windows, but this is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if the SRAM window overlaps a local access window.

See Section 7.3.1.4, "L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBARn)," for information about configuring SRAM windows.

2.2.2 Window into Configuration Space

CCSRBAR defines a window used to access all memory-mapped configuration, control, and status registers. No address translation is done, so there are no associated translation address registers. The window is always enabled with a fixed size of 1 Mbyte; no other attributes are attached, so there is no associated size/attribute register. This window always takes precedence over all local access windows. See Section 4.3.1.1.2, "Configuration, Control, and Status Base Address Register (CCSRBAR)," and Section 2.3, "Configuration, Control, and Status Register Map."

2.2.3 Local Access Windows

As demonstrated in the address map overview in Section 2.1, "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 MPC8555E 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, define its size, and specify the target interface.

With the exception of configuration space (mapped by CCSRBAR), on-chip SRAM regions (mapped by L2SRBAR registers), and default boot ROM, all addresses used by the system must be mapped by a local access window. This includes addresses that are mapped by inbound ATMU windows; target mappings of inbound ATMU windows and local access windows must be consistent.

The local access window registers exist as part of the local access block in the general utilities registers. See Section 2.3.4, "General Utilities 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.

2.2.3.1 Local Access Register Memory Map

Table 2-4 shows the memory map for the local access registers.

Table 2-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C10	LAWSR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.3.3/2-6

2.2.3.2 Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR7)

Figure 2-2 shows the bit fields of the LAWBAR*n* registers.

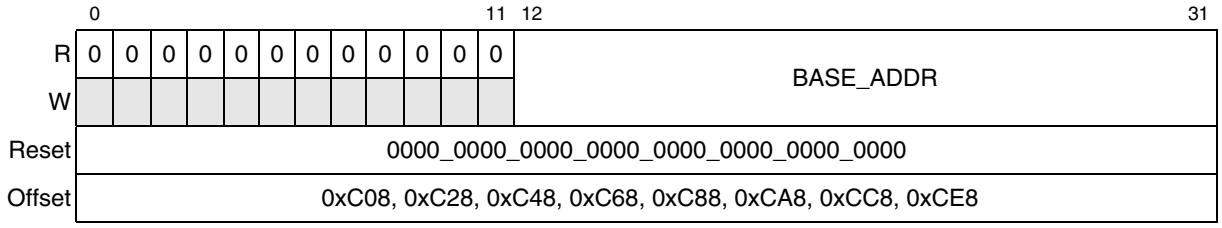


Figure 2-2. Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR7)

Table 2-5 describes LAWBAR*n* bit settings.

Table 2-5. LAWBAR*n* Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0
12–31	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window *n*. The specified base address should be aligned to the window size, as defined by LAWAR*n*[SIZE].

2.2.3.3 Local Access Window *n* Attributes Registers (LAWAR0–LAWAR7)

Figure 2-3 shows the bit fields of the LAWAR*n* registers.

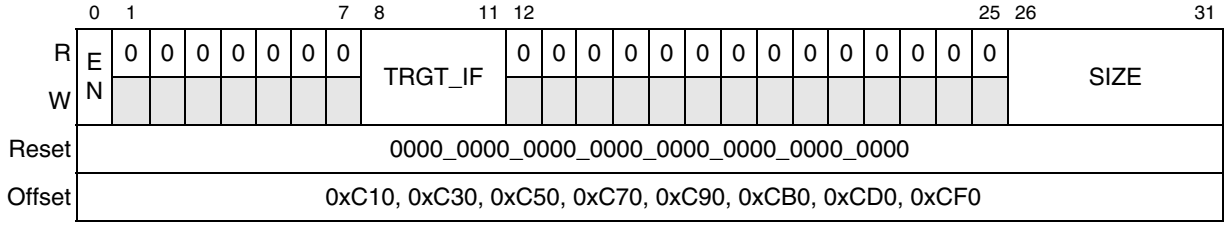


Figure 2-3. Local Access Window *n* Attributes Registers (LAWAR0–LAWAR7)

Table 2-6 describes LAWAR*n* bit settings.

Table 2-6. LAWAR*n* Bit Settings

Bits	Name	Description
0	EN	0 The local access window *n* (and all other LAWAR*n* and LAWBAR*n* fields) are disabled. 1 The local access window *n* is enabled and other LAWAR*n* and LAWBAR*n* fields combine to identify an address range for this window.
1–7	—	Write reserved, read = 0
8–11	TRGT_IF	Identifies the target interface ID when a transaction hits in the address range defined by this window. Note that configuration registers and SRAM regions are mapped by the windows defined by CCSRBAR and L2SRBAR. These mappings supersede local access window mappings, so configuration registers and SRAM do not appear as a target for local access windows. 0000 PCI 1 0001 PCI 2 0010–0011 Reserved 0100 Local bus memory controller 0101–1110 Reserved 1111 DDR SDRAM
12–25	—	Write reserved, read = 0
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes .. . . . . $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved

2.2.3.4 Precedence of Local Access Windows

If two local access windows overlap, the lower numbered window takes precedence. For instance, if two windows are setup as shown in Table 2-7, 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 2 also encompasses that memory region.

Table 2-7. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
1	0x7FF0_0000	1 Mbyte	0b0100 (local bus controller —LBC)
2	0x0000_0000	2 Gbytes	0b1111 (DDR SDRAM)

2.2.3.5 Configuring Local Access Windows

Once a local access window is enabled, it should not be modified while any device in the system may be using the window. Neither should a new window be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last local access window configuration register before enabling any other devices to use the window. For instance, if local access windows 0–3 are being configured in order during the initialization process, the last write (to LAWAR3) should be followed by a read of LAWAR3 before any devices try to use any of these windows. If the configuration is being done by the local e500 processor, the read of LAWAR3 should be followed by an **isync** instruction.

2.2.3.6 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 MPC8555E internal interconnects from the transactions source to its target. After 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. Similarly, the local bus controller has base registers that perform a similar function. The PCIs have interface has outbound address translation and mapping 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 ATMU windows. A single local access window can be further decoded to any number of chip selects or to any number of outbound ATMU windows at the target interface.

2.2.3.7 Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects

If a local access window maps an address to an interface other than the DDR SDRAM controller, then there should not be a valid chip select configured for the same address in the DDR SDRAM controller. Because

DDR SDRAM chip select boundaries are defined by a beginning and ending address, it is easy to define them so that they do not overlap with local access windows that map to other interfaces.

2.2.4 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 the MPC8555E, both the PCI blocks have outbound address translation and mapping units (ATMUs).

The PCI controllers have four outbound ATMU windows plus a default window. The PCI outbound ATMU registers include an extended translation address register so that up to 64 bits of external address space can be supported. See Section 16.3.1.2, "PCI ATMU Outbound Registers," for a detailed description of the PCI outbound ATMU windows.

2.2.5 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 to the local address space understood by the internal interfaces of the MPC8555E. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. Both the PCI controllers have inbound address translation and mapping units (ATMUs).

2.2.5.1 PCI Inbound ATMU

The PCI controller has three general inbound ATMU windows plus a dedicated window for memory mapped configuration accesses (PCSRBAR). 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 ATMU windows, the MPC8555E does not respond with an assertion of $\overline{\text{PCI}n\text{\_DEVSEL}}$. See Section 16.3.1.2, "PCI ATMU Outbound Registers," for a detailed description of the PCI inbound ATMU windows.

2.2.5.2 Illegal Interaction Between Inbound ATMUs and Local Access Windows

Since both local access windows and inbound ATMUs map transactions to a target interface, it is essential that they not contradict one another. For instance, it is a programming error to have an inbound ATMU map a transaction to the DDR SDRAM memory controller (target interface 0b1111) if the resulting translated local address is mapped to PCI1 (target interface 0b0000) by a local access window. Such a programming error may result in unpredictable system deadlocks.

2.3 Configuration, Control, and Status Register Map

All of the memory mapped configuration, control, and status registers in the MPC8555E are contained within a 1-Mbyte address region. To allow for flexibility, the configuration, control, and status block is relocatable in the local address space. The local address map location of this register block is controlled by the configuration, control, and status registers base address register (CCSRBAR), see Section 4.3.1.1.2, "Configuration, Control, and Status Base Address Register (CCSRBAR)." The default value for CCSRBAR is 4 Gbytes–9 Mbytes, or 0xFF70_0000.

NOTE

The configuration, control, and status window must not overlap a local access window that maps to the DDR controller. Otherwise, undefined behavior occurs.

An example of a top-level memory map with the default location of the configuration, control, and status registers is shown in Figure 2-4.

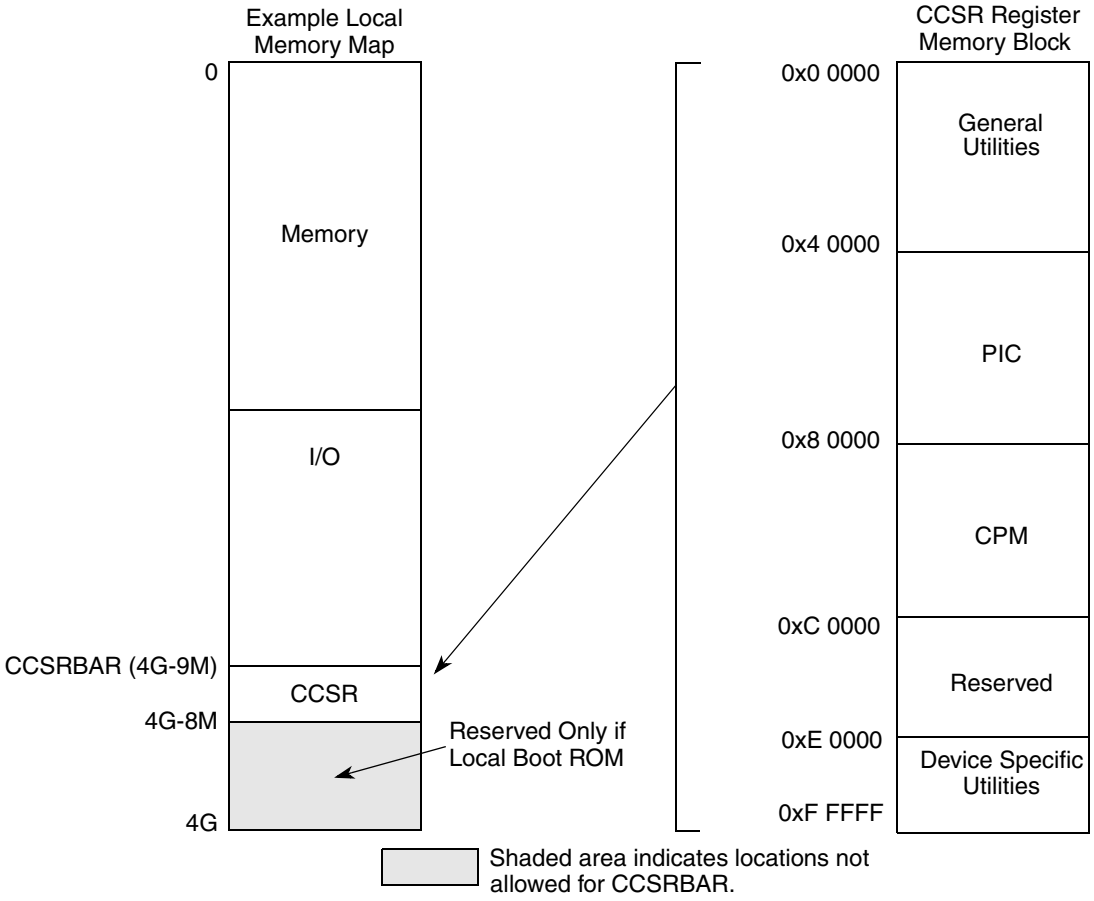


Figure 2-4. Top-Level Register Map Example

2.3.1 Accessing CCSR Memory from the e500 Core

When the local e500 processor is used to configure CCSR space, the CCSR memory space should typically be marked as Cache-Inhibited and Guarded.

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

To guarantee that the results of any sequence of writes to configuration registers are in effect, the final configuration register write should be 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.

2.3.2 Accessing CCSR Memory from External Masters

In addition to being accessible by the e500 processor, the configuration, control, and status registers are accessible from external interfaces. This allows external masters on the I/O ports to configure the MPC8555E.

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

The PCI base address for accessing the local CCSR memory is selectable through the PCI configuration and status register base address register (PCSRBAR), at offset 0x10, described in Section 16.3.2.11, "PCI Base Address Registers." An external PCI master sets this register by running a PCI configuration cycle to the MPC8555E. Subsequent memory accesses by a PCI master to the PCI address range indicated by PCSRBAR are translated to the local address indicated by the current setting of CCSRBAR.

2.3.3 Organization of CCSR Memory

The configuration, control, and status registers of the MPC8555E are grouped according to functional units. Most functional blocks are allocated a 4-Kbyte address space for registers. Registers that fall into this category are referred to as general utilities registers. These registers occupy the first 256 Kbytes of CCSR memory.

Registers that control functions that are not particular to a functional unit but to the device as a whole occupy the highest 256 Kbytes of CCSR memory. These are referred to as device-specific registers.

Some functional units, such as and the OpenPIC-based interrupt controller and the CPM have larger address spaces as defined by their programming models. The registers for these blocks are given their own large regions of CCSR memory.

Table 2-8. Local Memory Configuration, Control, and Status Register Summary

Offset from CCSRBAR	Register Grouping
0x0_0000–0x3_FFFF	General utilities
0x4_0000–0x7_FFFF	Programmable interrupt controller (PIC)
0x8_0000–0xB_FFFF	CPM
0xC_0000–0xD_FFFF	Reserved
0xE_0000–0xF_FFFF	Device-specific utilities

2.3.4 General Utilities Registers

Figure 2-5 provides an overview of the general utilities registers.

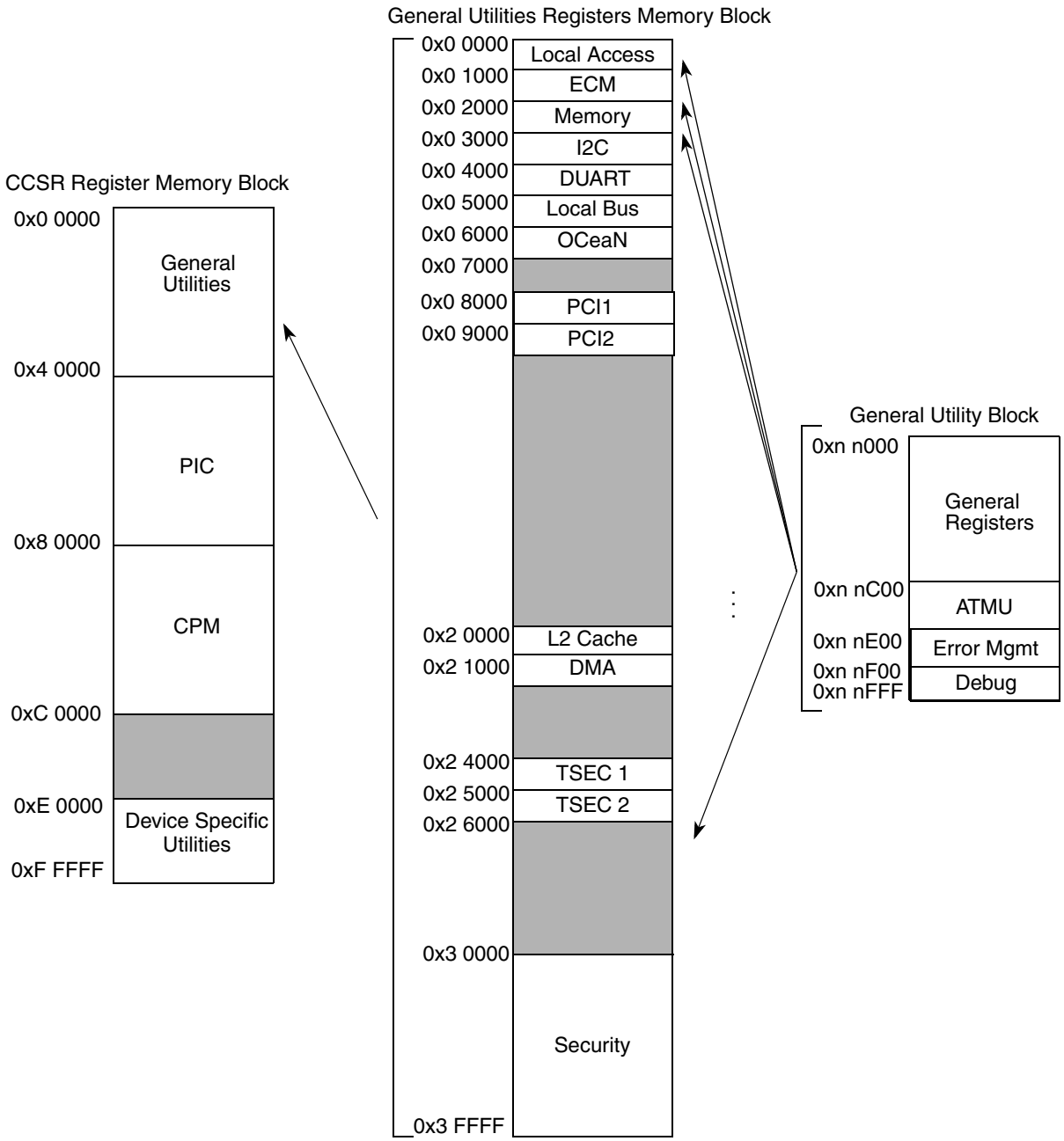


Figure 2-5. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block

Figure 2-5 also shows the organization of registers inside the 4-Kbyte register space allocated to an individual functional block. The first 3 Kbytes are available for general registers. The next 512 bytes are dedicated to address translation and mapping registers, if applicable to that particular functional unit (for example, PCI). If a unit has error management registers, they are typically placed starting at offset 0xE00

from the beginning of the block's 4-Kbyte space, and any debug registers are typically placed in the final 256 bytes of the unit's register space starting at offset 0xF00.

General utilities registers are accessed as 32-bit quantities except for the DUART and I^2C registers, which are accessed as bytes.

NOTE

Refer to detailed register descriptions for each functional unit for exact locations, sizes, and access requirements. Some blocks may have exceptions to the above guidelines.

2.3.5 Interrupt Controller and CCSR

The programmable interrupt controller (PIC) registers are at offset 0x4_0000 from CCSRBAR, see Figure 2-6. Its programming model follows the OpenPIC architecture. The interrupt controller registers should only be accessed with 32-bit accesses.

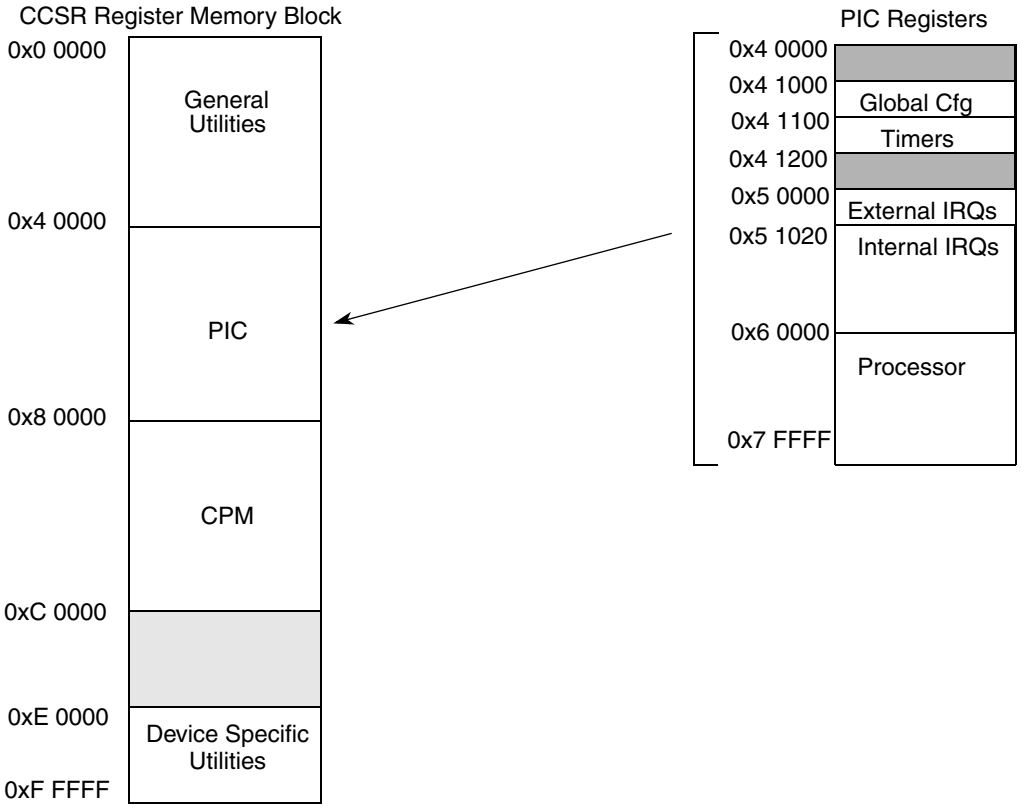


Figure 2-6. PIC Mapping to Configuration, Control, and Status Memory Block

2.3.6 Communications Processor Module and CCSR

The communication processor module (CPM) uses 256 Kbytes of configuration memory. In addition to a 64-Kbyte region for configuration registers, two separate 16-Kbyte parameter RAM regions are defined as well as a 32-Kbyte instruction RAM region.

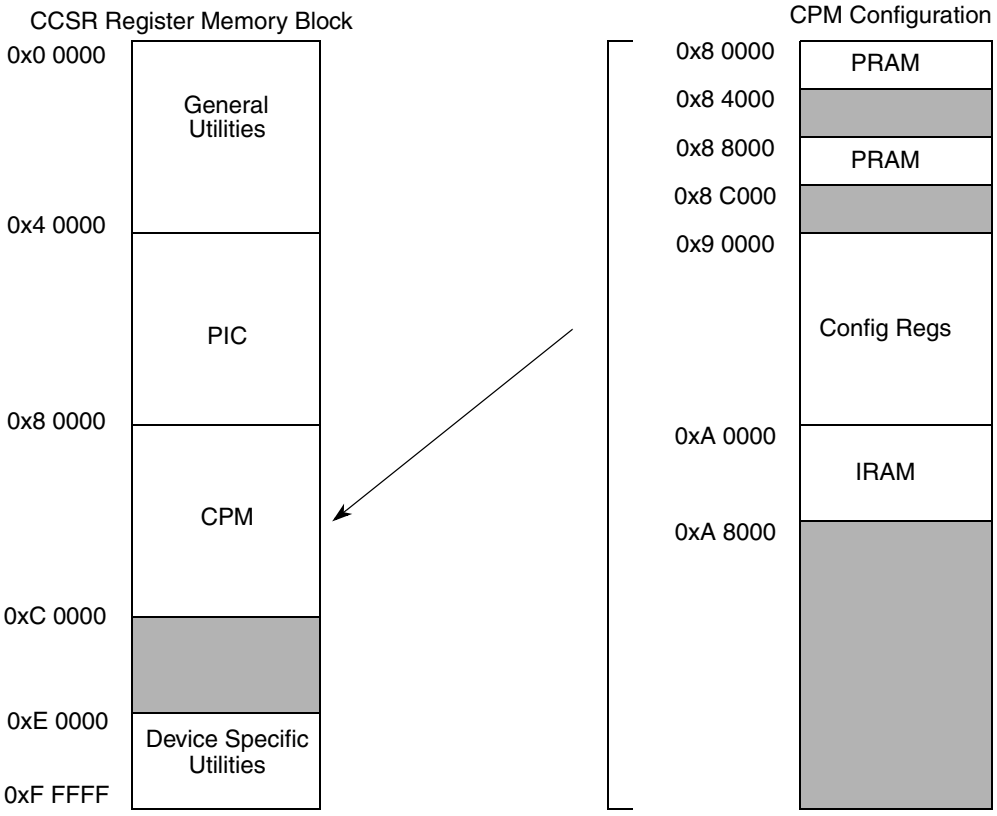


Figure 2-7. CPM Mapping to Configuration, Control, and Status Memory Block

2.3.7 Device-Specific Utilities

The device-specific registers consist of power management, performance monitors, and device-wide debug utilities (refer to Figure 2-8). These registers are accessible with 32-bit accesses only. Transactions of other than 32-bit are considered a programming error and operation is undefined.

Reserved bits in the following register descriptions are not guaranteed to have predictable values. Software must preserve the values of reserved bits when writing to a register. Also, when reading from a register, software should not rely on the value of any reserved bit remaining consistent.

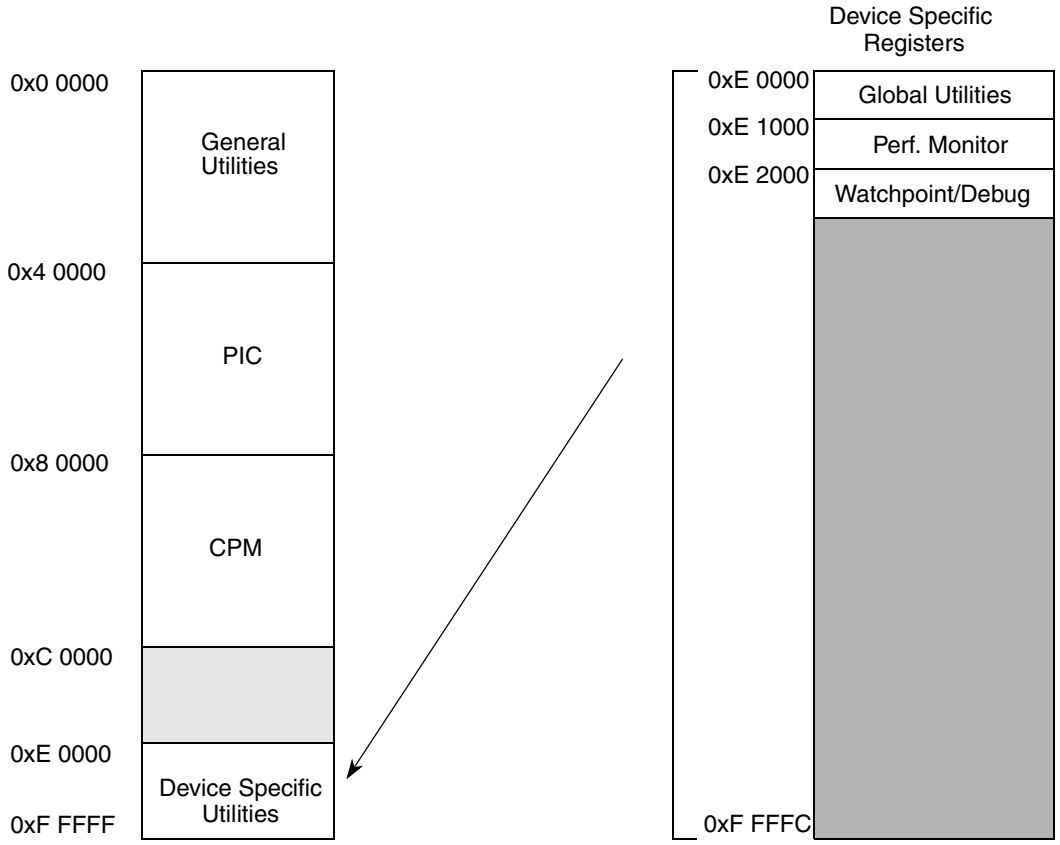


Figure 2-8. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block

2.4 Complete CCSR Map

Table 2-9 lists the MPC8555E memory-mapped registers.

Table 2-9. Memory Map

Offset	Register	Access	Reset	Section/Page
Local-Access Registers—Configuration, Control, and Status Registers				
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-5
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-6
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-7
Local-Access Registers—Local-Access Window Base and Size Registers				
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C10	LAWSR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.3.3/2-6

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.3.2/2-5
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
Registers				
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*00_0000	8.2.1.2/8-4
0x0_1E00	EEDR—ECM error detect register	Special	0x0000_0000	8.2.1.3/8-4
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.4/8-5
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.5/8-6
0x0_1E10	EEADR—ECM error address capture register	R	0x0000_0000	8.2.1.6/8-7
DDR Memory Controller Memory Map				
0x0_2000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-9
0x0_2008	CS1_BNDS—Chip select 1 memory bounds			
0x0_2010	CS2_BNDS—Chip select 2 memory bounds			
0x0_2018	CS3_BNDS—Chip select 3 memory bounds			
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-10
0x0_2084	CS1_CONFIG—Chip select 1 configuration			
0x0_2088	CS2_CONFIG—Chip select 2 configuration			
0x0_208C	CS3_CONFIG—Chip select 3 configuration			
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.3/9-11
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.4/9-12
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.5/9-13
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.6/9-14

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.7/9-15
0x0_2130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0000_0000	9.4.1.8/9-16
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.9/9-17
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.10/9-17
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.11/9-18
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.12/9-19
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.13/9-19
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.14/9-20
0x0_2E40	ERR_DETECT—Memory error detect	Special	0x0000_0000	9.4.1.15/9-20
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.16/9-21
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.17/9-22
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.18/9-22
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.19/9-23
0x0_2E58	ERR_SBE—Single-bit ECC memory error management	R/W	0x0000_0000	9.4.1.20/9-24
I^2C				
0x0_3000	I2CADR—I^2C address register	R/W	0x00	11.3.1.1/11-5
0x0_3004	I2CFDR—I^2C frequency divider register	R/W	0x00	11.3.1.2/11-6
0x0_3008	I2CCR—I^2C control register	R/W	0x00	11.3.1.3/11-7
0x0_300C	I2CSR—I^2C status register	R/W	0x81	11.3.1.4/11-9
0x0_3010	I2CDR—I^2C data register	R/W	0x00	11.3.1.5/11-10
0x0_3014	I2CDFSRR—I^2C digital filter sampling rate register	R/W	0x10	11.3.1.6/11-11
DUART Registers				
0x0_4500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	12.3.1.1/12-6
0x0_4500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	12.3.1.2/12-6
0x0_4500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x0_4501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	12.3.1.5/12-9
0x0_4502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	12.3.1.6/12-11

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_4502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	12.3.1.12/12-17
0x0_4503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	12.3.1.7/12-12
0x0_4504	UMCR—ULCR[DLAB] = x UART0 MODEM control register	R/W	0x00	12.3.1.8/12-14
0x0_4505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	12.3.1.9/12-14
0x0_4506	UMSR—ULCR[DLAB] = x UART0 MODEM status register	R	0x00	12.3.1.10/12-16
0x0_4507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	12.3.1.11/12-17
0x0_4510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	12.3.1.13/12-18
0x0_4600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	12.3.1.1/12-6
0x0_4600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	12.3.1.2/12-6
0x0_4600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x0_4601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	12.3.1.5/12-9
0x0_4602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	12.3.1.6/12-11
0x0_4602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	12.3.1.12/12-17
0x0_4603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	12.3.1.7/12-12
0x0_4604	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	0x00	12.3.1.8/12-14
0x0_4605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	12.3.1.9/12-14
0x0_4606	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	0x00	12.3.1.10/12-16
0x0_4607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	12.3.1.11/12-17
0x0_4610	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	12.3.1.13/12-18
Local Bus Controller Registers				
0x0_5000	BR0—Base register 0	R/W	0x0000_RR01[1]	13.3.1.1/13-10
0x0_5008	BR1—Base register 1		0x0000_0000	
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 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	13.3.1.2/13-12
0x0_500C	OR1—Options register 1		0x0000_0000	
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	0x0000_0000	13.3.1.3/13-17
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	13.3.1.4/13-18
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	13.3.1.4/13-18
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	13.3.1.4/13-18
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	13.3.1.5/13-20
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	13.3.1.6/13-21
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	13.3.1.7/13-21
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	13.3.1.8/13-23
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	13.3.1.9/13-24
0x0_50B0	LTESR—Transfer error status register	Read/ Bit-reset	0x0000_0000	13.3.1.10/13-25
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	13.3.1.11/13-26
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	13.3.1.12/13-27
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	13.3.1.13/13-28
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	13.3.1.14/13-29
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	13.3.1.15/13-29
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	13.3.1.16/13-31
PCI Registers				
PCI1 Configuration Access Registers				
0x0_8000	CFG_ADDR—PCI1 configuration address	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8004	CFG_DATA—PCI1 configuration data	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8008	INT_ACK—PCI1 interrupt acknowledge	R	0x0000_0000	16.3.1.1.3/16-19
0x0_800C–0x0_8BFC	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
PCI1 ATMU Registers—Outbound and Inbound				
0x0_8C00–0x0_8C3C–Outbound Window 0 (default)				
0x0_8C00	POTAR0—PCI1 outbound window 0 (default) translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C04	POTEAR0—PCI1 outbound window 0 (default) translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C08	Reserved	—	—	
0x0_8C0C	Reserved	—	—	
0x0_8C10	POWAR0—PCI1 outbound window 0 (default) attributes register	R/W	0x8004_401F	16.3.1.2.4/16-22
0x0_8C14–0x0_8C1C	Reserved	—	—	
0x0_8C20–0x0_8C3C—Outbound Window 1				
0x0_8C20	POTAR1—PCI1 outbound window 1 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C24	POTEAR1—PCI1 outbound window 1 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C28	POWBAR1—PCI1 outbound window 1 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C2C	Reserved	—	—	
0x0_8C30	POWAR1—PCI1 outbound window 1 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C34–0x0_8C3C	Reserved	—	—	
0x0_8C40–0x0_8C5C—Outbound Window 2				
0x0_8C40	POTAR2—PCI1 outbound window 2 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C44	POTEAR2—PCI1 outbound window 2 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C48	POWBAR2—PCI1 outbound window 2 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C4C	Reserved	—	—	
0x0_8C50	POWAR2—PCI1 outbound window 2 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C54–0x0_8C5C	Reserved	—	—	
0x0_8C60–0x0_8C7C—Outbound Window 3				
0x0_8C60	POTAR3—PCI1 outbound window 3 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C64	POTEAR3—PCI1 outbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C68	POWBAR3—PCI1 outbound window 3 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C6C	Reserved	—	—	
0x0_8C70	POWAR3—PCI1 outbound window 3 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C74–0x0_8C7C	Reserved	—	—	
0x0_8C80–0x0_8C9C—Outbound Window 4				
0x0_8C80	POTAR4—PCI1 outbound window 4 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C84	POTEAR4—PCI1 outbound window 4 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C88	POWBAR4—PCI1 outbound window 4 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C8C	Reserved	—	—	
0x0_8C90	POWAR4—PCI1 outbound window 4 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C94–0x0_8D9C	Reserved	—	—	
0x0_8DA0–0x0_8DBC–Inbound Window 3				
0x0_8DA0	PITAR3—PCI1 inbound window 3 translation address register	R/W	0x0000_0000	16.3.1.3.1/16-24
0x0_8DA4	Reserved	—	—	
0x0_8DA8	PIWBAR3—PCI1 inbound window 3 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DAC	PIWBEAR3—PCI1 inbound window 3 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DB0	PIWAR3—PCI1 inbound window 3 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DB4–0x0_8DBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC–Inbound Window 2				
0x0_8DC0	PITAR2—PCI1 inbound window 2 translation address register	R/W	0x0000_0000	16.3.1.3.1/16-24
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI1 inbound window 2 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DCC	PIWBEAR2—PCI1 inbound window 2 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DD0	PIWAR2—PCI1 inbound window 2 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DD4–0x0_8DDC	Reserved	—	—	
0x0_8DE0–0x0_8DFC–Inbound Window 1				
0x0_8DE0	PITAR1—PCI1 inbound window 1 translation address register	R/W	0x0000_0000	16.3.1.3.1/16-24
0x0_8DE4	Reserved	—	—	

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8DE8	PIWBAR1—PCI1 inbound window 1 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PIWAR1—PCI1 inbound window 1 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DF4–0x0_8DFC	Reserved	—	—	
PCI1 Error Management Registers				
0x0_8E00	ERR_DR—PCI1 error detect register	Special	0x0000_0000	16.3.1.4.1/16-29
0x0_8E04	ERR_CAP_DR—PCI1 error capture disabled register	R/W	0x0000_0000	16.3.1.4.2/16-30
0x0_8E08	ERR_EN—PCI1 error enable register	R/W	0x0000_0000	16.3.1.4.3/16-31
0x0_8E0C	ERR_ATTRIB—PCI1 error attributes capture register	R/W	0x0000_0000	16.3.1.4.4/16-32
0x0_8E10	ERR_ADDR—PCI1 error address capture register	R/W	0x0000_0000	16.3.1.4.5/16-33
0x0_8E14	ERR_EXT_ADDR—PCI1 error extended address capture register	R/W	0x0000_0000	16.3.1.4.6/16-33
0x0_8E18	ERR_DL—PCI1 error data low capture register	R/W	0x0000_0000	16.3.1.4.7/16-33
0x0_8E1C	ERR_DH—PCI1 error data high capture register	R/W	0x0000_0000	16.3.1.4.8/16-34
0x0_8E20	GAS_TIMR—PCI1 gasket timer register	R/W	0x0000_0000	16.3.1.4.9/16-35
0x0_8E24–0x0_8EFC	Reserved	—	—	
0x0_8F00–0x0_8FFC	Reserved for debug	—	—	
0x0_9000–0x0_9FFC	PCI2 Registers **Note:** The PCI2 Interface has the same memory-mapped registers that are described for PCI1 from 0x0_8000 to 0x0_8FFF except the offsets are from 0x0_9000 to 0x0_9FFF			
L2/SRAM Memory-Mapped Configuration Registers				
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	7.3.1.1/7-7
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-11

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E40	L2ERRDET—L2 error detect register	Special	0x0000_0000	7.3.1.5.2/7-14
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E50	L2ERRADDR—L2 error address capture register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.5.2/7-14
DMA Registers				
0x2_1100	MR*n*—DMA 0 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x2_1104	SR*n*—DMA 0 status register	Special	0x0000_0000	15.3.2.2/15-11
0x2_1108	Reserved	—	—	—
0x2_110C	CLNDAR*n*—DMA 0 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x2_1110	SATR*n*—DMA 0 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x2_1114	SAR*n*—DMA 0 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x2_1118	DATR*n*—DMA 0 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x2_111C	DAR*n*—DMA 0 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x2_1120	BCR*n*—DMA 0 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x2_1124	Reserved	—	—	—
0x2_1128	NLNDAR*n*—DMA 0 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x2_1130	Reserved	—	—	—
0x2_1134	CLSDAR*n*—DMA 0 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x2_1138	Reserved	—	—	—
0x2_113C	NLSDAR*n*—DMA 0 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x2_1140	SSR*n*—DMA 0 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x2_1144	DSR*n*—DMA 0 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1180	MR*n*—DMA 1 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x2_1184	SR*n*—DMA 1 status register	Special	0x0000_0000	15.3.2.2/15-11
0x2_1188	Reserved	—	—	—
0x2_118C	CLNDAR*n*—DMA 1 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x2_1190	SATR*n*—DMA 1 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x2_1194	SAR*n*—DMA 1 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x2_1198	DATR*n*—DMA 1 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x2_119C	DAR*n*—DMA 1 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x2_11A0	BCR*n*—DMA 1 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x2_11A4	Reserved	—	—	—
0x2_11A8	NLNDAR*n*—DMA 1 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x2_11B0	Reserved	—	—	—
0x2_11B4	CLSDAR*n*—DMA 1 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x2_11B8	Reserved	—	—	—
0x2_11BC	NLSDAR*n*—DMA 1 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x2_11C0	SSR*n*—DMA 1 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x2_11C4	DSR*n*—DMA 1 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x2_1200	MR*n*—DMA 2 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x2_1204	SR*n*—DMA 2 status register	Special	0x0000_0000	15.3.2.2/15-11
0x2_1208	Reserved	—	—	—
0x2_120C	CLNDAR*n*—DMA 2 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x2_1210	SATR*n*—DMA 2 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x2_1214	SAR*n*—DMA 2 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x2_1218	DATR*n*—DMA 2 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x2_121C	DAR*n*—DMA 2 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x2_1220	BCR*n*—DMA 2 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x2_1224	Reserved	—	—	—
0x2_1228	NLNDAR*n*—DMA 2 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x2_1230	Reserved	—	—	—
0x2_1234	CLSDAR*n*—DMA 2 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x2_1238	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_123C	NLSDAR*n*—DMA 2 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x2_1240	SSR*n*—DMA 2 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x2_1244	DSR*n*—DMA 2 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x2_1280	MR*n*—DMA 3 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x2_1284	SR*n*—DMA 3 status register	Special	0x0000_0000	15.3.2.2/15-11
0x2_1288	Reserved	—	—	—
0x2_128C	CLNDAR*n*—DMA 3 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x2_1290	SATR*n*—DMA 3 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x2_1294	SAR*n*—DMA 3 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x2_1298	DATR*n*—DMA 3 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x2_129C	DAR*n*—DMA 3 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x2_12A0	BCR*n*—DMA 3 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x2_12A4	Reserved	—	—	—
0x2_12A8	NLNDAR*n*—DMA 3 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x2_12B0	Reserved	—	—	—
0x2_12B4	CLSDAR*n*—DMA 3 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x2_12B8	Reserved	—	—	—
0x2_12BC	NLSDAR*n*—DMA 3 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x2_12C0	SSR*n*—DMA 3 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x2_12C4	DSR*n*—DMA 3 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x2_1300	DGSR—DMA general status register	Read	0x0000_0000	15.3.2.14/15-21
TSEC1 General Control and Status Registers				
0x2_4000–0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	14.5.3.1.1/14-19
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	14.5.3.1.2/14-22
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	14.5.3.1.3/14-24
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	14.5.3.1.4/14-25
0x2_4024	MINFLR—Minimum frame length register	R/W	0x0000_0040	14.5.3.1.5/14-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	14.5.3.1.6/14-26
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	14.5.3.1.7/14-27

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	14.5.3.1.8/14-28
0x2_4034–0x2_4088	Reserved	R	0x0000_0000	—
TSEC1 FIFO Control and Status Registers				
0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	14.5.3.2.1/14-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—
0x2_408C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	14.5.3.2.2/14-30
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	14.5.3.2.3/14-31
0x2_409C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	14.5.3.2.4/14-31
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	14.5.3.3.1/14-32
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	14.5.3.3.2/14-33
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TBDLEN—TxBD data length register	R	0x0000_0000	14.5.3.3.3/14-34
0x2_4110	TXIC—Transmit interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.3.4/14-34
0x2_4114–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	14.5.3.3.5/14-35
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TBPTR—TxBD pointer register	R/W	0x0000_0000	14.5.3.3.6/14-35
0x2_4188–0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TBASE—TxBD base address register	R/W	0x0000_0000	14.5.3.3.7/14-36
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	14.5.3.3.8/14-36
0x2_42B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	14.5.3.3.9/14-38
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
TSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	14.5.3.4.1/14-39
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	14.5.3.4.2/14-40
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RBDLEN—RxBD data length register	R	0x0000_0000	14.5.3.4.3/14-40
0x2_4310	RXIC—Receive interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.4.4/14-41
0x2_4314–0x2_4320	Reserved	R	0x0000_0000	—
0x2_4324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	14.5.3.4.5/14-42
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	14.5.3.4.6/14-42
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RBPTR—RxBD pointer register	R/W	0x0000_0000	14.5.3.4.7/14-43
0x2_4388–0x2_4400	Reserved	R	0x0000_0000	—
0x2_4404	RBASE—RxBD base address register	R/W	0x0000_0000	14.5.3.4.8/14-44
0x2_4408–0x2_44FC	Reserved	R	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register #1	R/W	0x0000_0000	14.5.3.6.1/14-47
0x2_4504	MACCFG2—MAC configuration register #2	R/W	0x0000_7000	14.5.3.6.2/14-48
0x2_4508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	14.5.3.6.3/14-49
0x2_450C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	14.5.3.6.4/14-50
0x2_4510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	14.5.3.6.5/14-51
0x2_4514–0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	14.5.3.6.6/14-52
0x2_4524	MIIMCOM—MII Management command register	R/W	0x0000_0000	14.5.3.6.7/14-53
0x2_4528	MIIMADD—MII Management address register	R/W	0x0000_0000	14.5.3.6.8/14-53
0x2_452C	MIIMCON—MII Management control register	W	0x0000_0000	14.5.3.6.9/14-54
0x2_4530	MIIMSTAT—MII Management status register	R	0x0000_0000	14.5.3.6.10/14-55
0x2_4534	MIIMIND—MII Management indicator register	R	0x0000_0000	14.5.3.6.11/14-55
0x2_4538	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_453C	IFSTAT—Interface status register	R	0x0000_0000	14.5.3.6.12/14-56
0x2_4540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	14.5.3.6.13/14-56
0x2_4544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	14.5.3.6.14/14-57
0x2_4548–0x2_467C	Reserved	R	0x0000_0000	—
TSEC1 RMON MIB Registers				
TSEC1 Transmit and Receive Counters				
0x2_4680	TR64—Transmit and receive 64-byte frame counter register	R/W	0x0000_0000	14.5.3.7.1/14-58
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter register	R/W	0x0000_0000	14.5.3.7.2/14-58
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter register	R/W	0x0000_0000	14.5.3.7.3/14-59
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter register	R/W	0x0000_0000	14.5.3.7.4/14-59
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter register	R/W	0x0000_0000	14.5.3.7.5/14-60
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter register	R/W	0x0000_0000	14.5.3.7.6/14-60
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count register	R/W	0x0000_0000	14.5.3.7.7/14-61
TSEC1 Receive Counters				
0x2_469C	RBYT—receive byte counter register	R/W	0x0000_0000	14.5.3.7.8/14-61
0x2_46A0	RPKT—receive packet counter register	R/W	0x0000_0000	14.5.3.7.9/14-62
0x2_46A4	RFCS—receive FCS error counter register	R/W	0x0000_0000	14.5.3.7.10/14-62
0x2_46A8	RMCA—receive multicast packet counter register	R/W	0x0000_0000	14.5.3.7.11/14-63
0x2_46AC	RBCA—receive broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.12/14-63
0x2_46B0	RXCF—receive control frame packet counter register	R/W	0x0000_0000	14.5.3.7.13/14-64
0x2_46B4	RXPF—receive PAUSE frame packet counter register	R/W	0x0000_0000	14.5.3.7.14/14-64
0x2_46B8	RXUO—receive unknown OP code counter register	R/W	0x0000_0000	14.5.3.7.15/14-65
0x2_46BC	RALN—receive alignment error counter register	R/W	0x0000_0000	14.5.3.7.16/14-65
0x2_46C0	RFLR—receive frame length error counter register	R/W	0x0000_0000	14.5.3.7.17/14-66
0x2_46C4	RCDE—receive code error counter register	R/W	0x0000_0000	14.5.3.7.18/14-66
0x2_46C8	RCSE—receive carrier sense error counter register	R/W	0x0000_0000	14.5.3.7.19/14-67
0x2_46CC	RUND—receive undersize packet counter register	R/W	0x0000_0000	14.5.3.7.20/14-67
0x2_46D0	ROVR—receive oversize packet counter register	R/W	0x0000_0000	14.5.3.7.21/14-68

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_46D4	RFRG—receive fragments counter register	R/W	0x0000_0000	14.5.3.7.22/14-68
0x2_46D8	RJBR—receive jabber counter register	R/W	0x0000_0000	14.5.3.7.23/14-69
0x2_46DC	RDRP—receive drop register	R/W	0x0000_0000	14.5.3.7.24/14-69
TSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter register	R/W	0x0000_0000	14.5.3.7.25/14-70
0x2_46E4	TPKT—Transmit packet counter register	R/W	0x0000_0000	14.5.3.7.26/14-70
0x2_46E8	TMCA—Transmit multicast packet counter register	R/W	0x0000_0000	14.5.3.7.27/14-71
0x2_46EC	TBCA—Transmit broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.28/14-71
0x2_46F0	TXPF—Transmit PAUSE control frame counter register	R/W	0x0000_0000	14.5.3.7.29/14-72
0x2_46F4	TDFR—Transmit deferral packet counter register	R/W	0x0000_0000	14.5.3.7.30/14-72
0x2_46F8	TEDF—Transmit excessive deferral packet counter register	R/W	0x0000_0000	14.5.3.7.31/14-73
0x2_46FC	TSCL—Transmit single collision packet counter register	R/W	0x0000_0000	14.5.3.7.32/14-73
0x2_4700	TMCL—Transmit multiple collision packet counter register	R/W	0x0000_0000	14.5.3.7.33/14-74
0x2_4704	TLCL—Transmit late collision packet counter register	R/W	0x0000_0000	14.5.3.7.34/14-74
0x2_4708	TXCL—Transmit excessive collision packet counter register	R/W	0x0000_0000	14.5.3.7.35/14-75
0x2_470C	TNCL—Transmit total collision counter register	R/W	0x0000_0000	14.5.3.7.36/14-75
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter register	R/W	0x0000_0000	14.5.3.7.37/14-76
0x2_4718	TJBR—Transmit jabber frame counter register	R/W	0x0000_0000	14.5.3.7.38/14-76
0x2_471C	TFCS—Transmit FCS error counter register	R/W	0x0000_0000	14.5.3.7.39/14-77
0x2_4720	TXCF—Transmit control frame counter register	R/W	0x0000_0000	14.5.3.7.40/14-77
0x2_4724	TOVR—Transmit oversize frame counter register	R/W	0x0000_0000	14.5.3.7.41/14-78
0x2_4728	TUND—Transmit undersize frame counter register	R/W	0x0000_0000	14.5.3.7.42/14-78
0x2_472C	TFRG—Transmit fragments frame counter register	R/W	0x0000_0000	14.5.3.7.43/14-79
TSEC1 General Registers				
0x2_4730	CAR1—Carry register one register	R/W	0x0000_0000	14.5.3.7.44/14-79
0x2_4734	CAR2—Carry register two register	R/W	0x0000_0000	14.5.3.7.45/14-80
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE01_FFFF	14.5.3.7.46/14-82
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFF	14.5.3.7.47/14-83
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
TSEC1 Hash Function Registers				
0x2_4800	IADDR0—Individual address register 0	R/W	0x0000_0000	14.5.3.8.1/14-84
0x2_4804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_4808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_480C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_4810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_4814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_4818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_481C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_4820– 0x2_487C	Reserved	R	0x0000_0000	—
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	14.5.3.8.2/14-85
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0– 0x2_4BF4	Reserved	R	0x0000_0000	—
TSEC1 Attribute Registers				
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	14.5.3.9.1/14-85
0x2_4BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	14.5.3.9.2/14-87
TSEC1 Future Expansion Space				
0x2_4C00– 0x2_4FFF	Reserved	R	0x0000_0000	—
TSEC2 Registers				
0x2_5000– 0x2_5FFF	TSEC2 registers **Note:** TSEC2 has the same memory-mapped registers that are described for TSEC1 from 0x 2_4000 to 0x2_4FFF except that the offsets are from 0x 2_5000 to 0x2_5FFF.			
Security Engine Register Address Map				
Controller Registers				
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000 _0000_0000	17.6.2.1/17-93

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1010	ISR—Interrupt status register	R	0x0000_0000 _0000_0000	17.6.2.2/17-94
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000 _0000_0000	17.6.2.3/17-95
0x3_1020	ID—Identification register	R	0x0000_0000 _0000_0040	17.6.2.4/17-97
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0 _00FF_F0F0	17.6.2/17-92
0x3_1030	MCR—Master control register	R/W	0x0000_0000 _0000_0000	17.6.2.5/17-98
Channel 1				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	0x0000_0000 _0000_0000	17.5.1.1/17-81
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000 _0000_0007	17.5.1.2/17-83
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	0x0000_0000 _0000_0000	17.5.1.3/17-89
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	0x0000_0000 _0000_0000	17.5.1.4/17-90
0x3_1180– 0x3_11BF	DBn—Crypto-channel 1descriptor buffers[0–7]	R	0x0000_0000 _0000_0000	17.5.1.5/17-90
Channel 2				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	0x0000_0000 _0000_0000	17.5.1.1/17-81
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000 _0000_0007	17.5.1.2/17-83
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	0x0000_0000 _0000_0000	17.5.1.3/17-89
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	0x0000_0000 _0000_0000	17.5.1.4/17-90
0x3_1280– 0x3_12BF	DBn—Crypto-channel 2 descriptor buffers[0–7]	R	0x0000_0000 _0000_0000	17.5.1.5/17-90
Channel 3				
0x3_1308	CCCR3—Crypto-channel 3 configuration register	R/W	0x0000_0000 _0000_0000	17.5.1.1/17-81
0x3_1310	CCPSR3—Crypto-channel 3 pointer status register	R	0x0000_0000 _0000_0007	17.5.1.2/17-83
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	0x0000_0000 _0000_0000	17.5.1.3/17-89

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	0x0000_0000 _0000_0000	17.5.1.4/17-90
0x3_1380–0x3_13BF	DBn—Crypto-channel 3 descriptor buffers[0–7]	R	0x0000_0000 _0000_0000	17.5.1.5/17-90
Channel 4				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	0x0000_0000 _0000_0000	17.5.1.1/17-81
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000 _0000_0007	17.5.1.2/17-83
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	0x0000_0000 _0000_0000	17.5.1.3/17-89
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	0x0000_0000 _0000_0000	17.5.1.4/17-90
0x3_1480–0x3_14BF	DBn—Crypto-channel 4 descriptor buffers[0–7]	R	0x0000_0000 _0000_0000	17.5.1.5/17-90
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000 _0000_0000	17.4.2.1/17-33
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000 _0000_0000	17.4.2.2/17-34
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000 _0000_0000	17.4.2.3/17-35
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000 _0000_0000	17.4.2.4/17-36
0x3_2028	DEUSR—DEU status register	R	0x0000_0000 _0000_0000	17.4.2.5/17-37
0x3_2030	DEUISR—DEU interrupt status register	R	0x0000_0000 _0000_0000	17.4.2.6/17-38
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000 _0000_3000	17.4.2.7/17-39
0x3_2050	DEUEUG—DEU EU-Go register	W	0x0000_0000 _0000_0000	17.4.2.8/17-41
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000 _0000_0000	17.4.2.9/17-41
0x3_2400	DEUK1—DEU key 1 register	W	—	17.4.2.10/17-41
0x3_2408	DEUK2—DEU key 2 register	W	—	17.4.2.10/17-41
0x3_2410	DEUK3—DEU key 3 register	W	—	17.4.2.10/17-41
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000 _0000_0000	17.4.2.11/17-42

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000 _0000_0000	17.4.6.1/17-67
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000 _0000_0000	17.4.6.2/17-69
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000 _0000_0000	17.4.6.3/17-70
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000 _0000_0000	17.4.6.4/17-70
0x3_4028	AESUSR—AESU status register	R	0x0000_0000 _0000_0000	17.4.6.5/17-71
0x3_4030	AESUISR—AESU interrupt status register	R	0x0000_0000 _0000_0000	17.4.6.6/17-72
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000 _0000_1000	17.4.6.7/17-73
0x3_4050	AESUEMR—AESU end of message register	W	0x0000_0000 _0000_0000	17.4.6.8/17-75
0x3_4100	AESU context memory registers	R/W	0x0000_0000 _0000_0000	17.4.6.9/17-75
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000 _0000_0000	17.4.6.9.5/17-79
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000 _0000_0000	17.4.6.9.6/17-80
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000 _0000_0000	17.4.4.1/17-51
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000 _0000_0000	17.4.4.3/17-53
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000 _0000_0000	17.4.4.4/17-54
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000 _0000_0000	17.4.4.5/17-54
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000 _0000_0000	17.4.4.6/17-55
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000 _0000_0000	17.4.4.7/17-56
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000 _0000_1000	17.4.4.8/17-57
0x3_6050	MDEUEUG—MDEU EU-Go register	W	0x0000_0000 _0000_0000	17.4.4.9/17-58

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000_0000_0000	17.4.4.10/17-59
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000_0000_0000	17.4.4.11/17-60
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000_0000_0000	17.4.4.12/17-60
Arc Four Execution Unit (AFEU)				
0x3_8000	AFEUMR—AFEU mode register	R/W	0x0000_0000_0000_0000	17.4.3.1/17-42
0x3_8008	AFEUKSR—AFEU key size register	R/W	0x0000_0000_0000_0000	17.4.3.3/17-43
0x3_8010	AFEUDSR—AFEU data size register	R/W	0x0000_0000_0000_0000	17.4.3.4/17-44
0x3_8018	AFEURCR—AFEU reset control register	R/W	0x0000_0000_0000_0000	17.4.3.5/17-45
0x3_8028	AFEUSR—AFEU status register	R	0x0000_0000_0000_0000	17.4.3.6/17-46
0x3_8030	AFEUISR—AFEU interrupt status register	R	0x0000_0000_0000_0000	17.4.3.7/17-47
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000_0000_1000	17.4.3.8/17-48
0x3_8050	AFEUEMR—AFEU end of message register	W	0x0000_0000_0000_0000	17.4.3.9/17-50
0x3_8100–0x3_81FF	AFEU context memory registers	R/W	0x0000_0000_0000_0000	17.4.3.10.1/17-50
0x3_8200	AFEU context memory pointers	R/W	0x0000_0000_0000_0000	17.4.3.10.2/17-51
0x3_8400	AFEUK0—AFEU key register 0	W	—	17.4.3.11/17-51
0x3_8408	AFEUK1—AFEU key register 1	W	—	17.4.3.11/17-51
0x3_8800–0x3_8FFF	AFEU FIFO	R/W	0x0000_0000_0000_0000	17.4.3.11.1/17-51
Random Number Generator (RNG)				
0x3_A000	RNGMR—RNG mode register	R/W	0x0000_0000_0000_0000	17.4.5.1/17-61
0x3_A010	RNGDSR—RNG data size register	R/W	0x0000_0000_0000_0000	17.4.5.2/17-62
0x3_A018	RNGRCR—RNG reset control register	R/W	0x0000_0000_0000_0000	17.4.5.3/17-63
0x3_A028	RNGSR—RNG status register	R	0x0000_0000_0000_0000	17.4.5.4/17-63

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_A030	RNGISR—RNG interrupt status register	R	0x0000_0000 _0000_0000	17.4.5.5/17-64
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000 _0000_1000	17.4.5.6/17-65
0x3_A050	RNGEUG—RNG EU-Go register	W	0x0000_0000 _0000_0000	17.4.5.7/17-66
0x3_A800–0x3_AFFF	RNG FIFO	R	0x0000_0000 _0000_0000	17.4.5.8/17-66
Public Key Execution Unit (PKEU)				
0x3_C000	PKEUMR—PKEU mode register	R/W	0x0000_0000 _0000_0000	17.4.1.1/17-25
0x3_C008	PKEUKSR—PKEU key size register	R/W	0x0000_0000 _0000_0000	17.4.1.2/17-26
0x3_C010	PKEUDSR—PKEU data size register	R/W	0x0000_0000 _0000_0000	17.4.1.3/17-26
0x3_C018	PKEURCR—PKEU reset control register	R/W	0x0000_0000 _0000_0000	17.4.1.5/17-28
0x3_C028	PKEUSR—PKEU status register	R	0x0000_0000 _0000_0000	17.4.1.6/17-29
0x3_C030	PKEUISR—PKEU interrupt status register	R	0x0000_0000 _0000_0000	17.4.1.7/17-30
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000 _0000_1000	17.4.1.8/17-31
0x3_C040	PKEUABS—PKEU AB size register	R/W	0x0000_0000 _0000_0000	17.4.1.3/17-26
0x3_C050	PKEUEUG—PKEU EU-Go	W	0x0000_0000 _0000_0000	17.4.1.9/17-32

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C200–0x3_C23F	PKEU parameter memory A0	R/W	0x0000_0000_0000_0000	17.4.1.10/17-32
0x3_C240–0x3_C27F	PKEU parameter memory A1	R/W	0x0000_0000_0000_0000	
0x3_C280–0x3_C2BF	PKEU parameter memory A2	R/W	0x0000_0000_0000_0000	
0x3_C2C0–0x3_C2FF	PKEU parameter memory A3	R/W	0x0000_0000_0000_0000	
0x3_C300–0x3_C33F	PKEU parameter memory B0	R/W	0x0000_0000_0000_0000	
0x3_C340–0x3_C37F	PKEU parameter memory B1	R/W	0x0000_0000_0000_0000	
0x3_C380–0x3_C3BF	PKEU parameter memory B2	R/W	0x0000_0000_0000_0000	
0x3_C3C0–0x3_C3FF	PKEU parameter memory B3	R/W	0x0000_0000_0000_0000	
0x3_C400–0x3_C4FF	PKEU parameter memory E	W	0x0000_0000_0000_0000	
0x3_C800–0x3_C8FF	PKEU parameter memory N	R/W	0x0000_0000_0000_0000	
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.7.1/10-37
0x4_0050	IPIDR1—IPI 1 dispatch register			
0x4_0060	IPIDR2—IPI 2 dispatch register			
0x4_0070	IPIDR3—IPI 3 dispatch register			
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.7.2/10-38
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.7.3/10-39
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.7.4/10-39
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.7.5/10-40
0x4_00C0–0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x0037_0002	10.3.1.1/10-15
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.2/10-15
0x4_1030	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.3/10-16
0x4_1090	PIR—Processor initialization register	R/W	0x0000_0000	10.3.1.4/10-16
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.5/10-17
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.6/10-18
0x4_10F0	TFRR—Timer frequency reporting register	R/W	0x0000_0000	10.3.2.1/10-19
0x4_1100	GTCCR0—Global timer 0 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_1110	GTBCR0—Global timer 0 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_1120	GTVPR0—Global timer 0 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_1130	GTDR0—Global timer 0 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_1140	GTCCR1—Global timer 1 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_1150	GTBCR1—Global timer 1 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_1160	GTVPR1—Global timer 1 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_1170	GTDR1—Global timer 1 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_1180	GTCCR2—Global timer 2 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_1190	GTBCR2—Global timer 2 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_11A0	GTVPR2—Global timer 2 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_11B0	GTDR2—Global timer 2 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_11C0	GTCCR3—Global timer 3 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_11D0	GTBCR3—Global timer 3 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_11E0	GTVPR3—Global timer 3 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_11F0	GTDR3—Global timer 3 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_1200–0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-22
0x4_1310	IRQSR0—$\overline{\text{IRQ\_OUT}}$ summary register 0	R	0x0000_0000	10.3.3.1/10-24
0x4_1320	IRQSR1—$\overline{\text{IRQ\_OUT}}$ summary register 1	R	0x0000_0000	10.3.3.2/10-25
0x4_1330	CISR0—Critical Interrupt summary register 0	R	0x0000_0000	10.3.3.3/10-26
0x4_1340	CISR1—Critical Interrupt summary register 1	R	0x0000_0000	10.3.3.4/10-26

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_13D0– 0x4_13F0	Reserved	—	—	—
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-28
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1440– 0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-29
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-29
0x4_1520– 0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0180–0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	10.3.6.4/10-33

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0550	IIDR126—Internal interrupt 26 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0600–0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1680–0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per-CPU Registers				
0x6_0000–0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—P0 IPI 0 dispatch register	W	all zeros	10.3.7.1/10-37
0x6_0050	IPIDR1—P0 IPI 1 dispatch register			
0x6_0060	IPIDR2—P0 IPI 2 dispatch register			
0x6_0070	IPIDR3—P0 IPI 3 dispatch register			

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x6_0080	CTPR0—P0 current task priority register	R/W	0x0000_000F	10.3.7.2/10-38
0x6_0090	WHOAMI0—P0 who am I register	R	all zeros	10.3.7.3/10-39
0x6_00A0	IACK0—P0 interrupt acknowledge register	R	all zeros	10.3.7.4/10-39
0x6_00B0	EOI0—P0 end of interrupt register	W	all zeros	10.3.7.5/10-40
CPM Dual-Port RAM				
0x8_0000–0x8_1FFF	DPRAM1—Dual-port RAM	R/W	—	21.4/21-28
0x8_2000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_9FFF	DPRAM2—Dual-port RAM	R/W	—	21.4/21-28
0x8_A000–0x8_FFFF	Reserved	—	—	—
e500 Core Interface				
0x9_0000	CEAR—CPM Error Address Register	R	0x0000_0000	21.2.3.1.1/21-18
0x9_0004	CEER—CPM Error Event Register	R/W	0x0000	21.2.3.1.2/21-19
0x9_0006	CEMR—CPM Error Mask Register	R/W	0x0000	21.2.3.1.3/21-20
SDMA				
0x9_0050	SMAER—System bus address error register	R	0x0000_0000	27.1.1/27-2
0x9_0054	Reserved	—	—	—
0x9_0058	SMEVR—System bus event register	R/W	0x0000_0000	27.1.2/27-2
0x9_005C	SMCTR—System bus control register	R/W	0x3800_0000	27.1.3/27-3
0x9_0060	LMAER—Local bus address error register	R	0x0000_0000	27.1.1/27-2
0x9_0064	Reserved	—	—	—
0x9_0068	LMEVR—Local bus event register	R/W	0x0000_0000	27.1.2/27-2
0x9_006C	LMCTR—Local bus control register	R/W	0x3800_0000	27.1.3/27-3
Interrupt Controller				
0x9_0C00	SICR—CPM interrupt configuration register	R/W	0x0000_0000	22.5.1/22-9
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIVEC—CPM interrupt vector register	R/W	0x0000_0000	22.5.1.5/22-14
0x9_0C08	SIPNR_H—CPM interrupt pending register (high)	R/W	0x0000_0000	22.5.1.3/22-11
0x9_0C0C	SIPNR_L—CPM interrupt pending register (low)	R/W	0x0000_0000	22.5.1.3/22-11
0x9_0C10	Reserved	—	—	—
0x9_0C14	SCPRR_H—CPM interrupt priority register (high)	R/W	0x0530_9770	22.5.1.2/22-10

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_0C18	SCPRR_L—CPM interrupt priority register (low)	R/W	0x0530_9770	22.5.1.2/22-10
0x9_0C1C	SIMR_H—CPM interrupt mask register (high)	R/W	0x0000_0000	22.5.1.4/22-12
0x9_0C20	SIMR_L—CPM interrupt mask register (low)	R/W	0x0000_0000	22.5.1.4/22-12
0x9_0C24	SIEXR—CPM external interrupt control register	R/W	0x0000_0000	22.5.1.6/22-15
0x9_0C28–0x9_0C7F	Reserved	—	—	—
Clock				
0x9_0C80	SCCR—System clock control register	R/W	0x0000_0000	25.1/25-2
Input/Output Port				
0x9_0D00	PDIRA—Port A data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D04	PPARA—Port A pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D08	PSORA—Port A special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D0C	PODRA—Port A open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D10	PDATA—Port A data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D14–0x9_0D1F	Reserved	—	—	—
0x9_0D20	PDIRB—Port B data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D24	PPARB—Port B pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D28	PSORB—Port B special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D2C	PODRB—Port B open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D30	PDATB—Port B data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	PDIRC—Port C data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D44	PPARC—Port C pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D48	PSORC—Port C special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D4C	PODRC—Port C open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D50	PDATC—Port C data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	PDIRD—Port D data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D64	PPARD—Port D pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D68	PSORD—Port D special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D6C	PODRD—Port D open drain register	R/W	0x0000_0000	45.2.1/45-1

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_0D70	PDATD—Port D data register	R/W	0x0000_0000	45.2.2/45-4
CPM Timers				
0x9_0D80	TGCR1—Timer 1 and timer 2 global configuration register	R/W	0x00	26.2.2/26-3
0x9_0D81	Reserved	—	—	—
0x9_0D84	TGCR2—Timer 3 and timer 4 global configuration register	R/W	0x00	26.2.2/26-3
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	TMR1—Timer 1 mode register	R/W	0x0000	26.2.3/26-5
0x9_0D92	TMR2—Timer 2 mode register	R/W	0x0000	26.2.3/26-5
0x9_0D94	TRR1—Timer 1 reference register	R/W	0x0000	26.2.4/26-7
0x9_0D96	TRR2—Timer 2 reference register	R/W	0x0000	26.2.4/26-7
0x9_0D98	TCR1—Timer 1 capture register	R/W	0x0000	26.2.5/26-7
0x9_0D9A	TCR2—Timer 2 capture register	R/W	0x0000	26.2.5/26-7
0x9_0D9C	TCN1—Timer 1 counter	R/W	0x0000	26.2.6/26-7
0x9_0D9E	TCN2—Timer 2 counter	R/W	0x0000	26.2.6/26-7
0x9_0DA0	TMR3—Timer 3 mode register	R/W	0x0000	26.2.3/26-5
0x9_0DA2	TMR4—Timer 4 mode register	R/W	0x0000	26.2.3/26-5
0x9_0DA4	TRR3—Timer 3 reference register	R/W	0x0000	26.2.4/26-7
0x9_0DA6	TRR4—Timer 4 reference register	R/W	0x0000	26.2.4/26-7
0x9_0DA8	TCR3—Timer 3 capture register	R/W	0x0000	26.2.5/26-7
0x9_0DAA	TCR4—Timer 4 capture register	R/W	0x0000	26.2.5/26-7
0x9_0DAC	TCN3—Timer 3 counter	R/W	0x0000	26.2.6/26-7
0x9_0DAE	TCN4—Timer 4 counter	R/W	0x0000	26.2.6/26-7
0x9_0DB0	TER1—Timer 1 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB2	TER2—Timer 2 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB4	TER3—Timer 3 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB6	TER4—Timer 4 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB8–0x9_12FF	Reserved	—	—	—
FCC1				
0x9_1300	GFMR1—FCC1 general mode register	R/W	0x0000_0000	37.2/37-3

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1304	FPSMR1—FCC1 protocol-specific mode register	R/W	0x0000_0000	(ATM) 41.13.3/41-84 (Ethernet) 40.18.2/40-20 (HDLC) 38.6/38-8
0x9_1308	FTODR1—FCC1 transmit on demand register	R/W	0x0000	37.6/37-8
0x9_130A	Reserved	—	—	—
0x9_130C	FDSR1—FCC1 data synchronization register	R/W	0x7E7E	37.5/37-7
0x9_130E	Reserved	—	—	—
0x9_1310	FCCE1—FCC1 event register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1312	Reserved	—	—	—
0x9_1314	FCCM1—FCC1 mask register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1316	Reserved	—	—	—
0x9_1318	FCCS1—FCC1 status register	R	0x00	38.10/38-16 (HDLC)
0x9_1319	Reserved	—	—	—
0x9_131C	FTIRR1_PHY0—FCC1 transmit internal rate registers for PHY0	R/W	0x00	41.13.5/41-87 (ATM)
0x9_131D	FTIRR1_PHY1—FCC1 transmit internal rate registers for PHY1	R/W	0x00	41.13.5/41-87 (ATM)
0x9_131E	FTIRR1_PHY2—FCC1 transmit internal rate registers for PHY2	R/W	0x00	41.13.5/41-87 (ATM)
0x9_131F	FTIRR1_PHY3—FCC1 transmit internal rate registers for PHY3	R/W	0x00	41.13.5/41-87 (ATM)
FCC2				
0x9_1320	GFMR2—FCC2 general mode register	R/W	0x0000_0000	37.2/37-3
0x9_1324	FPSMR2—FCC2 protocol-specific mode register	R/W	0x0000_0000	(ATM) 41.13.3/41-84 (Ethernet) 40.18.2/40-20 (HDLC) 38.6/38-8

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1328	FTODR2—FCC2 transmit on-demand register	R/W	0x0000	37.6/37-8
0x9_132A	Reserved	—	—	—
0x9_132C	FDSR2—FCC2 data synchronization register	R/W	0x7E7E	37.5/37-7
0x9_132E	Reserved	—	—	—
0x9_1330	FCCE2—FCC2 event register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1332	Reserved	—	—	—
0x9_1334	FCCM2—FCC2 mask register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1336	Reserved	—	—	—
0x9_1338	FCCS2—FCC2 status register	R	0x00	38.10/38-16 (HDLC)
0x9_1339–0x9_137F	Reserved	—	0x00	—
0x9_133C	FTIRR2_PHY0—FCC2 transmit internal rate registers for PHY0	R/W	0x00	41.13.5/41-87 (ATM)
0x9_133D	FTIRR2_PHY1—FCC2 transmit internal rate registers for PHY1	R/W	0x00	41.13.5/41-87 (ATM)
0x9_133E	FTIRR2_PHY2—FCC2 transmit internal rate registers for PHY2	R/W	0x00	41.13.5/41-87 (ATM)
0x9_133F	FTIRR2_PHY3—FCC2 transmit internal rate registers for PHY3	R/W	0x00	41.13.5/41-87 (ATM)
0x9_1340–0x9_137F	Reserved	—	—	—
FCC1 (continued)				
0x9_1380	FIRPER1—FCC1 internal rate port enable register	R/W	0x0000_0000	41.15.3/41-90
0x9_1384	FIRER1—FCC1 internal rate event register	R/W	0x0000_0000	41.15.4/41-91
0x9_1388	FIRSR1_HI—FCC1 internal rate selection register: HI	R/W	0x0000_0000	41.15.5/41-92
0x9_138C	FIRSR1_LO— FCC1 internal rate selection register: LO	R/W	0x0000_0000	41.15.5/41-92
0x9_1390	GFEMR1—General FCC1 expansion mode register	R/W	0x00	37.3/37-7
0x9_1391–0x9_139F	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
FCC2 (continued)				
0x9_13A0	FIRPER2—FCC2 internal rate port enable register	R/W	0x0000_0000	41.15.3/41-90
0x9_13A4	FIRER2—FCC2 internal rate event register	R/W	0x0000_0000	41.15.4/41-91
0x9_13A8	FIRSR2_HI—FCC2 internal rate selection register: HI	R/W	0x0000_0000	41.15.5/41-92
0x9_13AC	FIRSR2_LO—FCC2 internal rate selection register: LO	R/W	0x0000_0000	41.15.5/41-92
0x9_13B0	GFEMR2—General FCC2 expansion mode register	R/W	0x00	37.3/37-7
0x9_13B1– 0x9_15EF	Reserved	—	—	—
BRGs 5–8				
0x9_15F0	BRGC5—BRG5 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15F4	BRGC6—BRG6 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15F8	BRGC7—BRG7 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15FC	BRGC8—BRG8 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_1600– 0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I2MOD—I^2C mode register	R/W	0x00	44.4.1/44-6
0x9_1861	Reserved	—	—	—
0x9_1864	I2ADD—I^2C address register	R/W	0x00	44.4.2/44-7
0x9_1865	Reserved	—	—	—
0x9_1868	II2BRG—I^2C BRG register	R/W	0x00	44.4.3/44-7
0x9_1869	Reserved	—	—	—
0x9_186C	I2COM—I^2C command register	R/W	0x00	44.4.5/44-8
0x9_186D	Reserved	—	—	—
0x9_1870	I2CER—I^2C event register	R/W	0x00	44.4.4/44-7
0x9_1871	Reserved	—	—	—
0x9_1874	I2CMR—I^2C mask register	R/W	0x00	44.4.4/44-7
0x9_1875– 0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	CPCR—Communications processor command register	R/W	0x0000_0000	21.3.1/21-24
0x9_19C4	RCCR—CP configuration register	R/W	0x0000_0000	21.2.6/21-22
0x9_19C8– 0x9_19D5	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_19D6	RTER—CP timers event register	R/W	0x0000	21.5.4/21-35
0x9_19DA	RTMR—CP timers mask register	R/W	0x0000	21.5.4/21-35
0x9_19DC	RTSCR—CP time-stamp timer control register	R/W	0x0000	21.2.7/21-23
0x9_19DE	Reserved	—	—	—
0x9_19E0	RTSR—CP time-stamp register	R/W	0x0000_0000	21.2.8/21-24
BRGs 1–4				
0x9_19F0	BRGC1—BRG1 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19F4	BRGC2—BRG2 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19F8	BRGC3—BRG3 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19FC	BRGC4—BRG4 configuration register	R/W	0x0000_0000	25.2/25-3
SCC1				
0x9_1A00	GSMR_L1—SCC1 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A04	GSMR_H1—SCC1 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A08	PSMR1—SCC1 protocol-specific mode register	R/W	0x0000	28.2/28-3 29.16/29-12 (UART) 30.8/30-7 (HDLC) 31.11/31-9 (BISYNC) 32.9/32-8 (Transparent)
0x9_1A0A	Reserved	—	—	—
0x9_1A0C	TODR1—SCC1 transmit-on-demand register	R/W	0x0000	28.2.3/28-9
0x9_1A0E	DSR1—SCC1 data synchronization register	R/W	0x7E7E	28.2.2/28-8
0x9_1A10	SCCE1—SCC1 event register	R/W		29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A14	SCCM1—SCC1 mask register	R/W	0x0000	
0x9_1A16	Reserved	—	—	—
0x9_1A17	SCCS1—SCC1 status register	R/W	0x00	29.20/29-20 (UART) 30.12/30-13 (HDLC) 31.15/31-15 (BISYNC) 32.13/32-12 (Transparent)
0x9_1A18– 0x9_1A3F	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
SCC3				
0x9_1A40	GSMR_L3—SCC3 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A44	GSMR_H3—SCC3 general mode register	R/W	0x0000_0000	
0x9_1A48	PSMR3—SCC3 protocol-specific mode register	R/W	0x0000	28.2.1/28-8 29.16/29-12 (UART) 30.8/30-7 (HDLC) 31.11/31-9 (BISYNC) 32.9/32-8 (Transparent)
0x9_1A4A	Reserved	—	—	—
0x9_1A4C	TODR3—SCC3 transmit on demand register	R/W	0x0000	28.2.2/28-8
0x9_1A4E	DSR3—SCC3 data synchronization register	R/W	0x7E7E	28.2.2/28-8
0x9_1A50	SCCE3—SCC3 event register	R/W		29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A54	SCCM3—SCC3 mask register	R/W	0x0000	
0x9_1A56	Reserved	—	—	—
0x9_1A57	SCCS3—SCC3 status register	R/W	0x00	29.20/29-20 (UART) 30.12/30-13 (HDLC) 31.15/31-15 (BISYNC) 32.13/32-12 (Transparent)
0x9_1A58– 0x9_1A5F	Reserved	—	—	—
SCC4				
0x9_1A60	GSMR_L4—SCC4 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A64	GSMR_H4—SCC4 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A68	PSMR4—SCC4 protocol-specific mode register	R/W	0x0000	28.2.1/28-8 29.16/29-12 (UART) 30.8/30-7 (HDLC) 31.11/31-9 (BISYNC) 32.9/32-8 (Transparent)
0x9_1A6A	Reserved	—	—	—
0x9_1A6C	TODR4—SCC4 transmit on-demand register	R/W	0x0000	28.2.3/28-9
0x9_1A6E	DSR4—SCC4 data synchronization register	R/W	0x7E7E	28.2.2/28-8

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1A70	SCCE4—SCC4 event register	R/W		29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A74	SCCM4—SCC4 mask register	R/W	0x0000	
0x9_1A76	Reserved	—	—	—
0x9_1A77	SCCS4—SCC4 status register	—	—	29.20/29-20 (UART) 30.12/30-13 (HDLC) 31.15/31-15 (BISYNC) 32.13/32-12 (Transparent)
0x9_1A78– 0x9_1A7F	Reserved	—	—	—
SMC1				
0x9_1A82	SMCMR1—SMC1 mode register	R/W	0x0000	36.2.1/36-2
0x9_1A84	Reserved	—	16 bits	—
0x9_1A86	SMCE1—SMC1 event register	R/W	0x00	36.3.11/36-18
0x9_1A87	Reserved	—	24 bits	—
0x9_1A8A	SMCM1—SMC1 mask register	R/W	0x00	36.3.11/36-18
0x9_1A8B– 0x9_1A91	Reserved	—	7 bytes	—
SMC2				
0x9_1A92	SMCMR2—SMC2 mode register	R/W	0x0000	36.2.1/36-2
0x9_1A94	Reserved	—	16 bits	—
0x9_1A96	SMCE2—SMC2 event register	R/W	0x00	36.3.11/36-18
0x9_1A97	Reserved	—	24 bits	—
0x9_1A9A	SMCM2—SMC2 mask register	R/W	0x00	36.3.11/36-18
0x9_1A9B– 0x9_1A9F	Reserved	—	5 bytes	—
SPI				
0x9_1AA0	SPMODE—SPI mode register	R/W	0x0000	43.4.1/43-6
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPIE—SPI event register	R/W	0x00	43.4.2/43-9
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPIM—SPI mask register	R/W	0x00	43.4.2/43-9
0x9_1AAB	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1AAD	SPCOM—SPI command register	W	0x00	43.4.3/43-10
0x9_1AA7–0x9_1B00	Reserved	—	—	—
CPM Mux				
0x9_1B02	CMXSI2CR—CPM mux SI2 clock route register	R/W	0x00	24.4.2/24-8
0x9_1B03	Reserved	—	—	—
0x9_1B04	CMXFCR—CPM mux FCC clock route register	R/W	0x0000_0000	24.4.3/24-8
0x9_1B08	CMXSCR—CPM mux SCC clock route register	R/W	0x0000_0000	24.4.4/24-10
0x9_1B0C	CMXSMR—CPM mux SMC clock route register	R/W	0x00	24.4.5/24-13
0x9_1B0E	CMXUAR—CPM mux UTOPIA address register	R/W	0x0000	24.4.1/24-5
0x9_1B10–0x9_1B3F	Reserved	—	—	—
SI1 Registers				
SI2 Registers				
0x9_1B40	SI2AMR—SI2 TDMA2 mode register	R/W	0x0000	23.6.2/23-14
0x9_1B42	SI2BMR—SI2 TDMB2 mode register	R/W	0x0000	23.6.2/23-14
0x9_1B44	SI2CMR—SI2 TDMC2 mode register	R/W	0x0000	23.6.2/23-14
0x9_1B46	Reserved	R/W	16 bits	23.6.2/23-14
0x9_1B48	SI2GMR—SI2 global mode register	R/W	0x00	23.6.1/23-14
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2CMDR—SI2 command register	R/W	0x00	23.6.4/23-20
0x9_1B4B	Reserved	—	—	—
0x9_1B4C	SI2STR—SI2 status register	R/W	0x00	23.6.5/23-21
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2RSR—SI2 RAM shadow address register	R/W	0x0000	23.6.3/23-20
0x9_1B50–0x9_1B5F	Reserved	—	—	—
USB				
0x9_1B60	USMOD—USB mode register	R/W	0x00	35.5.7.1/35-17
0x9_1B61	USADR—USB address register	R/W	0x00	35.5.7.2/35-18
0x9_1B62	USCOM—USB command register	R/W	0x00	35.5.7.4/35-19
0x9_1B64	USEP1—USB end point 1 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B66	USEP2—USB end point 2 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B68	USEP3—USB end point 3 register	R/W	0x0000	35.5.7.3/35-18

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1B6A	USEP4—USB end point 4 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B6C–0x9_1B6F	Reserved	—	32 bits	—
0x9_1B70	USBER—USB event register	R/W	0x0000	35.5.7.5/35-20
0x9_1B72	Reserved	—	16 bits	—
0x9_1B74	USBMR—USB mask register	R/W	0x0000	35.5.7.6/35-21
0x9_1B77	USBS—USB status register	R/W	0x00	35.5.7.7/35-21
0x9_1B79–0x9_1FFF	Reserved	—	1174 bytes	—
SI2 RAM				
0x9_2800–0x9_29FF	SI2TxRAM—SI 2 transmit routing RAM	—	—	23.5.3/23-9
0x9_2A00–0x9_2BFF	Reserved	—	—	—
0x9_2C00–0x9_2DFF	SI2RxRAM—SI 2 receive routing RAM	—	—	23.5.3/23-9
0x9_2E00–0x9_3FFF	Reserved	—	—	—
Instruction RAM				
0xA_0000–0xA_0FFF	Dual-port RAM (instruction RAM only)	—	—	21.4/21-28
Global Utilities Registers				
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_*n*1*nn*	18.4.1.1/18-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	18.4.1.2/18-5
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	R/W	0x000*n*_007F	18.4.1.3/18-6
0xE_000C	PORDEVSR—POR I/O device status register	R	see ref.	18.4.1.4/18-7
0xE_0010	PORDBGMSR—POR debug mode status register	R	see ref.	18.4.1.5/18-8
0xE_0020	GPPORCR—General-purpose POR configuration register	R	see ref.	18.4.1.6/18-9
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	18.4.1.7/18-10
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	18.4.1.8/18-11
0xE_0050	GPINDR—General-purpose input data register	R	0x*nnnn*_0000	18.4.1.9/18-12
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	18.4.1.9/18-12

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Device Disables				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	18.4.1.11/18-14
Power Management Registers				
0xE_0080	POWMGTCSR—Power management status and control register	R/W	0x0000_0000	18.4.1.12/18-16
Interrupt Reporting				
0xE_0090	MCPSUMR—Machine check summary register	Read/Clear	0x0000_0000	18.4.1.13/18-17
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	18.4.1.14/18-18
0xE_00A4	SVR—System version register	R	MPC8555E/ MPC8541E system version	18.4.1.15/18-19
Debug Control				
0xE_0E00	CLKOCR—Clock out select register	R/W	0x0000_0000	18.4.1.16/18-19
0xE_0E20	LBDLLCR—LBC DLL control register	R/W	0x0000_0000	18.4.1.17/18-20
Performance Monitor Control Registers				
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	19.3.2.1/19-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1018	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	19.3.3.1/19-9
0xE_101C	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1034	PMLCB2—Performance monitor local control register B2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	19.3.3.1/19-9

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1064	PMLCB5—Performance monitor local control register B5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	19.3.3.1/19-9
Debug and Watchpoint Monitor Registers				
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	20.3.1.1/20-10
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	20.3.1.1/20-10
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	20.3.1.2/20-12
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	20.3.1.3/20-13
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	20.3.1.4/20-13
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	20.3.1.5/20-15
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register	R/W	0x0000_0000	20.3.2.1/20-15
0xE_2044	TBCR1—Trace buffer control register	R/W	0x0000_0000	20.3.2.1/20-15
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	20.3.2.2/20-18
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	20.3.2.3/20-18
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	20.3.2.4/20-18
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	20.3.2.5/20-19
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	20.3.2.6/20-20
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	20.3.2.7/20-21
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	20.3.2.8/20-21
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	20.3.3.1/20-22
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	20.3.3.2/20-22

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	20.3.4.1/20-23

[1] Port size for BR0 is configured from external pins during reset, hence ‘RR’ is either 0x08, 0x10, or 0x18.

Chapter 3
Signal Descriptions

This chapter describes the MPC8555E 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

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 MPC8555E signals are grouped as follows:

- DDR memory interface signals
- PCI interface signals
- Ethernet management interface signals
- TSEC1 interface signals
- TSEC2 interface signals
- Local bus interface signals
- DMA interface signals
- PIC interface signals
- DUART interface signals
- I^2C interface signals
- CPM interface signals
- System control, power management, and debug signals
- Test, JTAG, and configuration signals
- Clock signals

Figure 3-1 illustrates the external signals of the MPC8555E, showing how the signals are grouped. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

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.

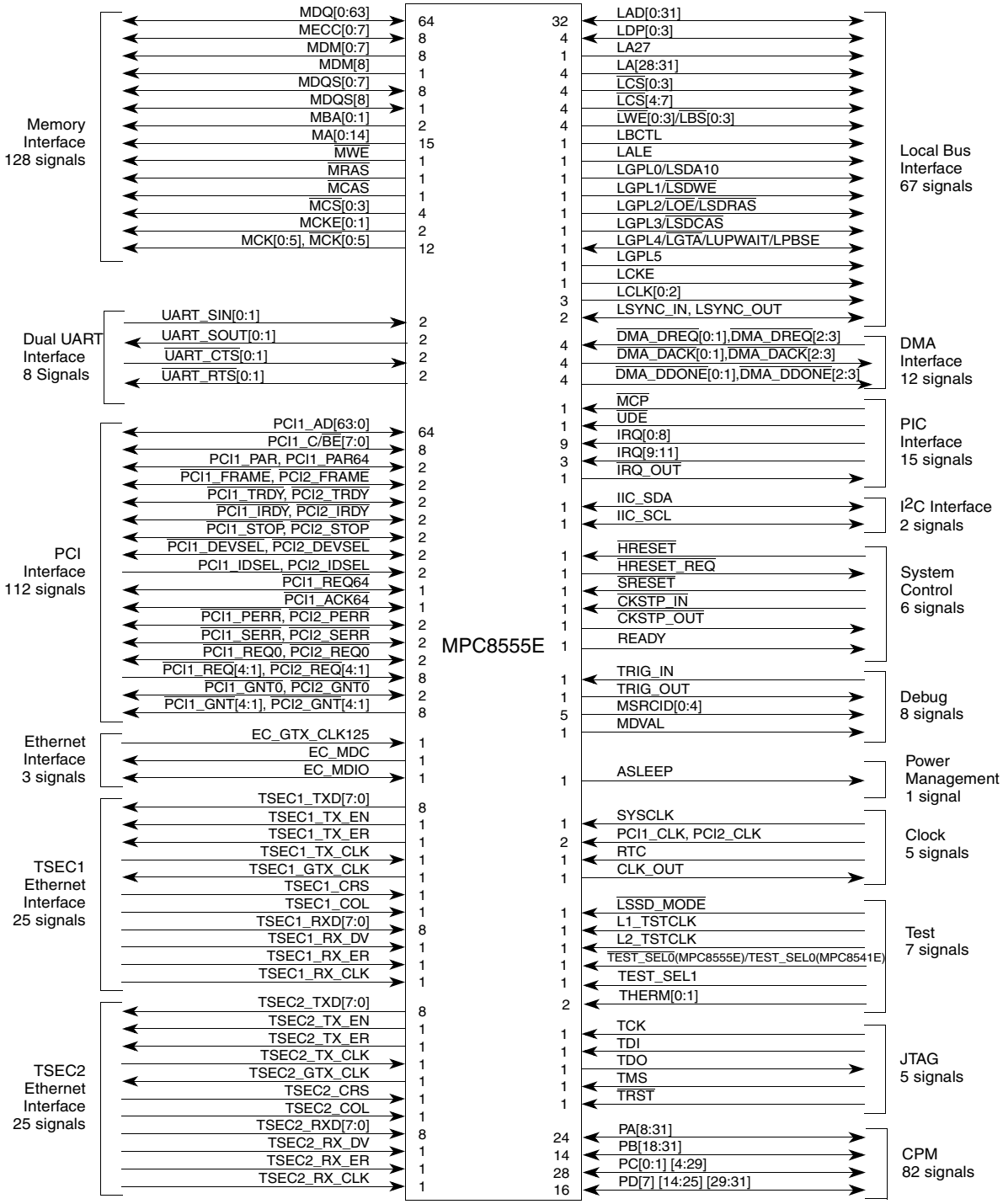


Figure 3-1. MPC8555E Signal Groupings

The following tables provide summaries of signal functions. Table 3-1 provides a summary of the signals grouped by function, and Table 3-4 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, output, or bidirectional. The direction of the multiplexed signals applies for the primary signal function listed in the left-most column of the table for that row (and does not apply for the state of the reset configuration signals). Finally, the table provides a pointer to the table where the signal function is described.

Table 3-1. MPC8555E Signal Reference by Functional Block

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
MDQ[0:63]	DDR data	DDR memory	—	64	I/O	9-3/9-5
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	9-3/9-5
MDM[0:7]	DDR data mask	DDR memory	—	8	O	9-3/9-5
MDM8	DDR ECC data mask	DDR memory	—	1	O	9-3/9-5
MDQS[0:7]	DDR data strobe	DDR memory	—	8	I/O	9-3/9-5
MDQS8	DDR ECC data strobe	DDR memory	—	1	I/O	9-3/9-5
MBA[0:1]	DDR bank select	DDR memory	—	2	O	9-3/9-5
MA[0:14]	DDR address	DDR memory	—	15	O	9-3/9-5
$\overline{\text{MWE}}$	DDR write enable	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MRAS}}$	DDR row address strobe	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MCAS}}$	DDR column address strobe	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR memory	—	4	O	9-3/9-5
MCKE[0:1]	DDR clock enable	DDR memory	—	2	O	9-4/9-8
MCK[0:5], $\overline{\text{MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	9-4/9-8
PCI1_AD[63:0]	PCI address/data	PCI	—	64	I/O	16-2/16-8
PCI1_C/$\overline{\text{BE}}$[7:0]	PCI command/byte enable	PCI	—	8	I/O	16-2/16-8
PCI1_PAR	PCI parity	PCI	—	1	I/O	16-2/16-8
PCI1_PAR64/ PCI2_PAR	PCI Parity 64	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_FRAME}}$	PCI frame	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_TRDY}}$, $\overline{\text{PCI2\_TRDY}}$	PCI target ready	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_IRDY}}$, $\overline{\text{PCI2\_IRDY}}$	PCI initiator ready	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_STOP}}$, $\overline{\text{PCI2\_STOP}}$	PCI stop	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_DEVSEL}}$	PCI device select	PCI	—	1	I/O	16-2/16-8

Table 3-1. MPC8555E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
PCI1_IDSEL, PCI2_IDSEL	PCI initial device select	PCI	—	2	I	16-2/16-8
$\overline{\text{PCI1\_REQ64}}$/ $\overline{\text{PCI2\_FRAME}}$	PCI Request 64/ PCI2 Frame	PCI	cfg_pci1_impd	1	I/O	16-2/16-8
$\overline{\text{PCI1\_ACK64}}$/ $\overline{\text{PCI2\_DEVSEL}}$	PCI Acknowledge 64/ PCI2 Device Select	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_PERR}}$, $\overline{\text{PCI2\_PERR}}$	PCI parity error	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_SERR}}$, $\overline{\text{PCI2\_SERR}}$	PCI system error	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_REQ}}$[4:0], $\overline{\text{PCI2\_REQ}}$[4:0]	PCI request 4–0	PCI	—	10	I	16-2/16-8
$\overline{\text{PCI1\_GNT0}}$	PCI1 grant 0	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_GNT1}}$	PCI1 grant 1	PCI	cfg_pci1_impd	1	O	16-2/16-8
$\overline{\text{PCI1\_GNT2}}$	PCI1 grant 2	PCI	cfg_pci1_arb	1	O	16-2/16-8
$\overline{\text{PCI1\_GNT3}}$	PCI1 grant 3	PCI	cfg_pci1_debug	1	O	16-2/16-8
$\overline{\text{PCI1\_GNT4}}$	PCI1 grant 4	PCI	cfg_pci1_hold_en	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT0}}$	PCI2 Grant 0	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI2\_GNT1}}$	PCI Grant 1	PCI	cfg_pc2_impd	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT2}}$	PCI Grant 2	PCI	cfg_pci2_arb	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT3}}$	PCI Grant 3	PCI	—	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT4}}$	PCI Grant 4	PCI	cfg_pci2_hold_en	1	O	16-2/16-8
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	14-3/14-14
EC_MDC	Ethernet management data clock	Ethernet management	cfg_tsec_reduce	1	O	14-3/14-14
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	14-3/14-14
TSEC1_TXD[7:0]	TSEC1 transmit data 7–0	TSEC1	—	8	O	14-3/14-14
TSEC1_TX_EN	TSEC1 transmit enable	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_ER	TSEC1 transmit error	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_CLK	TSEC1 transmit clock in	TSEC1	—	1	I	14-3/14-14
TSEC1_GTX_CLK	TSEC1 transmit clock out	TSEC1	_	1	O	14-3/14-14
TSEC1_CRS	TSEC1 carrier sense	TSEC1	—	1	I	14-3/14-14
TSEC1_COL	TSEC1 collision detect	TSEC1	—	1	I	14-3/14-14
TSEC1_RXD[7:0]	TSEC1 receive data 7–0	TSEC1	—	8	I	14-3/14-14

Table 3-1. MPC8555E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC1_RX_DV	TSEC1 receive data valid	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_ER	TSEC1 receiver error	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_CLK	TSEC1 receive clock	TSEC1	—	1	I	14-3/14-14
TSEC2_TXD[7:4]	TSEC2 transmit data 7–4	TSEC2	—	4	O	14-3/14-14
TSEC2_TXD3	TSEC2 transmit data 3	TSEC2	cfg_tsec1	1	O	14-3/14-14
TSEC2_TXD2	TSEC2 transmit data 2	TSEC2	cfg_tsec2	1	O	14-3/14-14
TSEC2_TXD1	TSEC2 Transmit Data 1	TSEC2	cfg_pci1_clk	1	O	14-3/14-14
TSEC2_TXD0	TSEC2 transmit data 0	TSEC2	cfg_pci2_clk	1	O	14-3/14-14
TSEC2_TX_EN	TSEC2 transmit enable	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_ER	TSEC2 transmit error	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_CLK	TSEC2 transmit clock in	TSEC2	—	1	I	14-3/14-14
TSEC2_GTX_CLK	TSEC2 transmit clock out	TSEC2	—	1	O	14-3/14-14
TSEC2_CRS	TSEC2 carrier sense	TSEC2	—	1	I	14-3/14-14
TSEC2_COL	TSEC2 collision detect	TSEC2	—	1	I	14-3/14-14
TSEC2_RXD[7:0]	TSEC2 receive data	TSEC2	—	8	I	14-3/14-14
TSEC2_RX_DV	TSEC2 receive data valid	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_ER	TSEC2 receiver error	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_CLK	TSEC2 receive clock	TSEC2	—	1	I	14-3/14-14
LAD[0:31]	LBC address/data	LBC	cfg_gpinput[0:31]	32	I/O	13-1/13-4
LDP[0:3]	LBC data parity	LBC	—	4	I/O	13-1/13-4
LA27	LBC burst address	LBC	cfg_cpu_boot	1	O	13-1/13-4
LA[28:31]	LBC port address	LBC	cfg_sys_pll[0:3]	4	O	13-1/13-4
$\overline{\text{LCS}}$[0:4]	LBC chip select 0–4	LBC	—	5	O	13-1/13-4
$\overline{\text{LCS5}}$	LBC chip select 5	LBC	$\overline{\text{DMA\_DREQ2}}$	1	O	13-1/13-4
$\overline{\text{LCS6}}$	LBC chip select 6	LBC	$\overline{\text{DMA\_DACK2}}$	1	O	13-1/13-4
$\overline{\text{LCS7}}$	LBC chip select 7	LBC	$\overline{\text{DMA\_DDONE2}}$	1	O	13-1/13-4
$\overline{\text{LWE0}}$/ LSDDQM0/LBS0	LBC write enable/byte lane data mask/byte select 0	LBC	cfg_lb_hold_en0	1	O	13-1/13-4
$\overline{\text{LWE1}}$/ LSDDQM1/LBS1	LBC write enable/byte lane data mask/byte select 1	LBC	cfg_lb_hold_en1	1	O	13-1/13-4
$\overline{\text{LWE2}}$/ LSDDQM2/LBS2	LBC write enable/byte lane data mask/byte select 2	LBC	cfg_host_agt	1	O	13-1/13-4
$\overline{\text{LWE3}}$/ LSDDQM3/LBS3	LBC write enable/byte lane data mask/byte select 3	LBC	cfg_rom_loc2	1	O	13-1/13-4

Table 3-1. MPC8555E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
LBCTL	LBC data buffer control	LBC	—	1	O	13-1/13-4
LALE	LBC address latch enable	LBC	cfg_core_pll0	1	O	13-1/13-4
LGPL0/LSDA10	LBC UPM general purpose line 0/SDRAM address bit 10	LBC	cfg_rom_loc0	1	O	13-1/13-4
LGPL1/$\overline{\text{LSDWE}}$	LBC GP line 1/SDRAM write enable	LBC	cfg_rom_loc1	1	O	13-1/13-4
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$	LBC GP line 2/output enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	13-1/13-4
LGPL3/$\overline{\text{LSDCAS}}$	LBC GP line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	13-1/13-4
LGPL4/$\overline{\text{LGTA}}$/ LUPWAIT/LPBSE	LBC GP line 4/GPCM terminate access/UPM wait/parity byte select	LBC	—	1	I/O	13-1/13-4
LGPL5	LBC GP line 5 address	LBC	cfg_boot_seq1	1	O	13-1/13-4
LCKE	LBC clock enable	LBC	—	1	O	13-1/13-4
LCLK[0:2]	LBC clock	LBC	—	3	O	13-1/13-4
LSYNC_IN, LSYNC_OUT	LBC DLL synchronization	LBC	—	2	I/O	13-1/13-4
$\overline{\text{DMA\_DREQ}}$[0:1]	DMA request 0–1	DMA	—	2	I	15-3/15-5
$\overline{\text{DMA\_DREQ2}}$	DMA request 2	DMA	$\overline{\text{LCS5}}$	1	I	15-3/15-5
$\overline{\text{DMA\_DREQ3}}$	DMA request 3	DMA	IRQ9	1	I	15-3/15-5
$\overline{\text{DMA\_DACK}}$[0:1]	DMA acknowledge 0–1	DMA	—	2	O	15-3/15-5
$\overline{\text{DMA\_DACK2}}$	DMA acknowledge 2	DMA	$\overline{\text{LCS6}}$	1	O	15-3/15-5
$\overline{\text{DMA\_DACK3}}$	DMA acknowledge 3	DMA	IRQ10	1	O	15-3/15-5
$\overline{\text{DMA\_DDONE}}$[0:1]	DMA done 0–1	DMA	—	2	O	15-3/15-5
$\overline{\text{DMA\_DDONE2}}$	DMA done 2	DMA	$\overline{\text{LCS7}}$	1	O	15-3/15-5
$\overline{\text{DMA\_DDONE3}}$	DMA done 3	DMA	IRQ11	1	O	15-3/15-5
$\overline{\text{MCP}}$	Machine check processor	PIC	—	1	I	10-5/10-7
$\overline{\text{UDE}}$	Unconditional debug event	PIC	—	1	I	10-5/10-7
IRQ[0:8]	External interrupt 0–8	PIC	—	9	I	10-5/10-7
IRQ9	External interrupt 9	PIC	$\overline{\text{DMA\_DREQ3}}$	1	I	10-5/10-7
IRQ10	External interrupt 10	PIC	$\overline{\text{DMA\_DACK3}}$	1	I	10-5/10-7
IRQ11	External interrupt 11	PIC	$\overline{\text{DMA\_DDONE3}}$	1	I	10-5/10-7
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-5/10-7
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	12-2/12-3

Table 3-1. MPC8555E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
UART_SOUT[0:1]	DUART serial data out	Dual UART	—	2	O	12-2/12-3
$\overline{\text{UART\_CTS}}$[0:1]	DUART clear to send	Dual UART	—	2	I	12-2/12-3
$\overline{\text{UART\_RTS}}$[0:1]	DUART ready to send	Dual UART	—	2	O	12-2/12-3
IIC_SDA	I^2C serial data	I^2C	—	1	I/O	11-2/11-4
IIC_SCL	I^2C serial clock	I^2C	—	1	I/O	11-2/11-4
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-2/4-2
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-2/4-2
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	18-2/18-2
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	18-2/18-2
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	20-2/20-5
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	20-2/20-5
MSRCID[0:1]	Memory debug source port ID 0–1	Debug	cfg_mem_debug cfg_ddr_debug	2	O	9-6/9-10
MSRCID[2:4]	Memory debug source port ID2–4	Debug	—	3	O	20-2/20-5
MDVAL	Memory debug data valid	Debug	—	1	O	20-2/20-5
ASLEEP	Asleep	Power mgmt	—	1	O	18-2/18-2
SYSCLK	System clock/PCI clock	Clock	—	1	I	4-3/4-3
PCI1_CLK, PCI2_CLK	Asynchronous PCI clock	Clock	—	2	I	4-3/4-3
RTC	Real time clock	Clock	—	1	I	4-3/4-3
CLK_OUT	Clock out	Clock	—	1	O	18-2/18-2
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	20-2/20-5
L1_TSTCLK	L1 test clock	Test	—	1	I	20-2/20-5
L2_TSTCLK	L2 test clock	Test	—	1	I	20-2/20-5
$\overline{\text{TEST\_SEL0}}$ (MPC8555E) TEST_SEL0 (MPC8541E)	Test select 0	Test	—	1	I	20-2/20-5
TEST_SEL1	Test select 1	Test	—	1	I	20-2/20-5
THERM[0:1]	Thermal resistor access	Test	—	2	I	20-2/20-5
TCK	Test clock	JTAG	—	1	I	20-2/20-5

Table 3-1. MPC8555E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TDI	Test data in	JTAG	—	1	I	20-2/20-5
TDO	Test data out	JTAG	—	1	O	20-2/20-5
TMS	Test mode select	JTAG	—	1	I	20-2/20-5
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	20-2/20-5
PA[8:31]	Port A	CPM	—	24	I/O	45-17/45-18
PB[18:31]	Port B	CPM	—	14	I/O	45-18/45-21
PC[0:1] [4:29]	Port C	CPM	—	28	I/O	45-19/45-22
PD[7] [14:25] [29:31]	Port D	CPM	—	16	I/O	45-20/45-24

Table 3-2. MPC8555E Alphabetical Signal Reference

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
ASLEEP	Asleep	Power mgmt	—	1	O	18-2/18-2
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	18-2/18-2
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	18-2/18-2
CLK_OUT	Clock out	Clock	—	1	O	18-2/18-2
$\overline{\text{DMA\_DACK2}}$	DMA acknowledge 2	DMA	$\overline{\text{LCS6}}$	1	O	15-3/15-5
$\overline{\text{DMA\_DACK3}}$	DMA acknowledge 3	DMA	IRQ10	1	O	15-3/15-5
$\overline{\text{DMA\_DACK}}$[0:1]	DMA acknowledge 0–1	DMA	—	2	O	15-3/15-5
$\overline{\text{DMA\_DDONE2}}$	DMA done 2	DMA	$\overline{\text{LCS7}}$	1	O	15-3/15-5
$\overline{\text{DMA\_DDONE3}}$	DMA done 3	DMA	IRQ11	1	O	15-3/15-5
$\overline{\text{DMA\_DDONE}}$[0:1]	DMA done 0–1	DMA	—	2	O	15-3/15-5
$\overline{\text{DMA\_DREQ2}}$	DMA request 2	DMA	$\overline{\text{LCS5}}$	1	I	15-3/15-5
$\overline{\text{DMA\_DREQ3}}$	DMA request 3	DMA	IRQ9	1	I	15-3/15-5
$\overline{\text{DMA\_DREQ}}$[0:1]	DMA request 0–1	DMA	—	2	I	15-3/15-5
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	14-3/14-14
EC_MDC	Ethernet management data clock	Ethernet management	cfg_tsec_reduce	1	O	14-3/14-14
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	14-3/14-14
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-2/4-2
IIC_SCL	I^2C serial clock	I^2C	—	1	I/O	11-2/11-4

Table 3-2. MPC8555E Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
IIC_SDA	I^2C serial data	I^2C	—	1	I/O	11-2/11-4
IRQ[0:8]	External interrupt 0–8	PIC	—	9	I	10-5/10-7
IRQ10	External interrupt 10	PIC	$\overline{\text{DMA\_DACK3}}$	1	I	10-5/10-7
IRQ11	External interrupt 11	PIC	$\overline{\text{DMA\_DDONE3}}$	1	I	10-5/10-7
IRQ9	External interrupt 9	PIC	$\overline{\text{DMA\_DREQ3}}$	1	I	10-5/10-7
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-5/10-7
L1_TSTCLK	L1 test clock	Test	—	1	I	20-2/20-5
L2_TSTCLK	L2 test clock	Test	—	1	I	20-2/20-5
LA27	LBC burst address	LBC	cfg_cpu_boot	1	O	13-1/13-4
LAD[0:31]	LBC address/data	LBC	cfg_gpinput[0:31]	32	I/O	13-1/13-4
LALE	LBC address latch enable	LBC	cfg_core_pll0	1	O	13-1/13-4
LA[28:31]	LBC port address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	13-1/13-4
LBCTL	LBC data buffer control	LBC	—	1	O	13-1/13-4
LCKE	LBC clock enable	LBC	—	1	O	13-1/13-4
LCLK[0:2]	LBC clock	LBC	—	3	O	13-1/13-4
$\overline{\text{LCS5}}$	LBC chip select 5	LBC	$\overline{\text{DMA\_DREQ2}}$	1	O	13-1/13-4
$\overline{\text{LCS6}}$	LBC chip select 6	LBC	$\overline{\text{DMA\_DACK2}}$	1	O	13-1/13-4
$\overline{\text{LCS7}}$	LBC chip select 7	LBC	$\overline{\text{DMA\_DDONE2}}$	1	O	13-1/13-4
$\overline{\text{LCS}}$[0:4]	LBC chip select 0–4	LBC	—	5	O	13-1/13-4
LDP[0:3]	LBC data parity	LBC	—	4	I/O	13-1/13-4
LGPL0/LSDA10	LBC UPM general purpose line 0/SDRAM address bit 10	LBC	cfg_rom_loc0	1	O	13-1/13-4
LGPL1/$\overline{\text{LSDWE}}$	LBC GP line 1/SDRAM write enable	LBC	cfg_rom_loc1	1	O	13-1/13-4
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$	LBC GP line 2/output enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	13-1/13-4
LGPL3/$\overline{\text{LSDCAS}}$	LBC GP line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	13-1/13-4
LGPL4/$\overline{\text{LGTA}}$/ LUPWAIT/LPBSE	LBC GP line 4/GPCM terminate access/UPM wait/parity byte select	LBC	—	1	I/O	13-1/13-4
LGPL5	LBC GP line 5 address	LBC	cfg_boot_seq1	1	O	13-1/13-4
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	20-2/20-5

Table 3-2. MPC8555E Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
LSYNC_IN, LSYNC_OUT	LBC DLL synchronization	LBC	—	2	I/O	13-1/13-4
$\overline{\text{LWE0}}$/ LSDDQM0/LBS0	LBC write enable/byte lane data mask/byte select 0	LBC	cfg_lb_hold_en0	1	O	13-1/13-4
$\overline{\text{LWE1}}$/ LSDDQM1/LBS1	LBC write enable/byte lane data mask/byte select 1	LBC	cfg_lb_hold_en1	1	O	13-1/13-4
$\overline{\text{LWE2}}$/ LSDDQM2/LBS2	LBC write enable/byte lane data mask/byte select 2	LBC	cfg_host_agt	1	O	13-1/13-4
$\overline{\text{LWE3}}$/ LSDDQM3/LBS3	LBC write enable/byte lane data mask/byte select 3	LBC	cfg_rom_loc2	1	O	13-1/13-4
MA[0:14]	DDR address	DDR memory	—	15	O	9-3/9-5
MBA[0:1]	DDR bank select	DDR memory	—	2	O	9-3/9-5
$\overline{\text{MCAS}}$	DDR column address strobe	DDR memory	—	1	O	9-3/9-5
MCKE[0:1]	DDR clock enable	DDR memory	—	2	O	9-4/9-8
MCK[0:5], $\overline{\text{MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	9-4/9-8
$\overline{\text{MCP}}$	Machine check processor	PIC	—	1	I	10-5/10-7
$\overline{\text{MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR memory	—	4	O	9-3/9-5
MDM8	DDR ECC data mask	DDR memory	—	1	O	9-3/9-5
MDM[0:7]	DDR data mask	DDR memory	—	8	O	9-3/9-5
MDQS8	DDR ECC data strobe	DDR memory	—	1	I/O	9-3/9-5
MDQS[0:7]	DDR data strobe	DDR memory	—	8	I/O	9-3/9-5
MDQ[0:63]	DDR data	DDR memory	—	64	I/O	9-3/9-5
MDVAL	Memory debug data valid	Debug	—	1	O	20-2/20-5
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	9-3/9-5
$\overline{\text{MRAS}}$	DDR row address strobe	DDR memory	—	1	O	9-3/9-5
MSRCID[0:1]	Memory debug source port ID 0–1	Debug	cfg_mem_debug cfg_ddr_debug	2	O	9-6/9-10
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	20-2/20-5
$\overline{\text{MWE}}$	DDR write enable	DDR memory	—	1	O	9-3/9-5
PA[8:31]	Port A	CPM	—	24	I/O	45-17/45-18
PB[18:31]	Port B	CPM	—	14	I/O	45-18/45-21
$\overline{\text{PCI1\_ACK64}}$/ $\overline{\text{PCI2\_DEVSEL}}$	PCI Acknowledge 64/ PCI2 Device Select	PCI	—	1	I/O	16-2/16-8
PCI1_AD[63:0]	PCI address/data	PCI	—	64	I/O	16-2/16-8

Table 3-2. MPC8555E Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
PCI1_C/$\overline{\text{BE}}$[7:0]	PCI command/byte enable	PCI	—	8	I/O	16-2/16-8
PCI1_CLK, PCI2_CLK	Asynchronous PCI clock	Clock	—	2	I	4-3/4-3
$\overline{\text{PCI1\_DEVSEL}}$	PCI device select	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_FRAME}}$	PCI1 frame	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_GNT0}}$	PCI1 grant 0	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_GNT1}}$	PCI1 grant 1	PCI	cfg_pci1_impd	1	O	16-2/16-8
$\overline{\text{PCI1\_GNT2}}$	PCI1 grant 2	PCI	cfg_pci1_arb	1	O	16-2/16-8
$\overline{\text{PCI1\_GNT3}}$	PCI1 grant 3	PCI	cfg_pci1_debug	1	O	16-2/16-8
$\overline{\text{PCI1\_GNT4}}$	PCI1 grant 4	PCI	cfg_pci1_hold_en	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT0}}$	PCI2 Grant 0	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI2\_GNT1}}$	PCI2 Grant 1	PCI	cfg_pci2_impd	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT2}}$	PCI2 Grant 2	PCI	cfg_pci2_arb	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT3}}$	PCI2 Grant 3	PCI	—	1	O	16-2/16-8
$\overline{\text{PCI2\_GNT4}}$	PCI2 Grant 4	PCI	cfg_pci2_hold_en	1	O	16-2/16-8
PCI1_IDSEL, PCI2_IDSEL	PCI initial device select	PCI	—	2	I	16-2/16-8
$\overline{\text{PCI1\_IRDY}}$, $\overline{\text{PCI2\_IRDY}}$	PCI initiator ready	PCI	—	2	I/O	16-2/16-8
PCI1_PAR	PCI1 parity	PCI	—	1	I/O	16-2/16-8
PCI1_PAR64/ PCI2_PAR	PCI1 Parity 64/PCI2 Parity	PCI	—	1	I/O	16-2/16-8
$\overline{\text{PCI1\_PERR}}$, $\overline{\text{PCI2\_PERR}}$	PCI parity error	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_REQ64}}$/ $\overline{\text{PCI2\_FRAME}}$	PCI Request 64/PCI Frame	PCI	cfg_pci1_impd	1	I/O	16-2/16-8
$\overline{\text{PCI1\_REQ}}$[4:0], $\overline{\text{PCI2\_REQ}}$[4:0]	PCI request 4–0	PCI	—	10	I	16-2/16-8
$\overline{\text{PCI1\_SERR}}$, $\overline{\text{PCI2\_SERR}}$	PCI system error	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_STOP}}$, $\overline{\text{PCI2\_STOP}}$	PCI stop	PCI	—	2	I/O	16-2/16-8
$\overline{\text{PCI1\_TRDY}}$, $\overline{\text{PCI2\_TRDY}}$	PCI target ready	PCI	—	2	I/O	16-2/16-8
PC[0:1] [4:29]	Port C	CPM	—	28	I/O	45-19/45-22

Table 3-2. MPC8555E Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
PD[7] [14:25] [29:31]	Port D	CPM	—	16	I/O	45-20/45-24
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
RTC	Real time clock	Clock	—	1	I	4-3/4-3
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-2/4-2
SYSCLK	System clock/PCI clock	Clock	—	1	I	4-3/4-3
TCK	Test clock	JTAG	—	1	I	20-2/20-5
TDI	Test data in	JTAG	—	1	I	20-2/20-5
TDO	Test data out	JTAG	—	1	O	20-2/20-5
$\overline{\text{TEST\_SEL0}}$ (MPC8555E) TEST_SEL0 (MPC8541E)	Test select 0	Test	—	1	I	20-2/20-5
TEST_SEL1	Test select 1	Test	—	1	I	20-2/20-5
THERM[0:1]	Thermal resistor access	Test	—	2	I	20-2/20-5
TMS	Test mode select	JTAG	—	1	I	20-2/20-5
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	20-2/20-5
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	20-2/20-5
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	20-2/20-5
TSEC1_COL	TSEC1 collision detect	TSEC1	—	1	I	14-3/14-14
TSEC1_CRS	TSEC1 carrier sense	TSEC1	—	1	I	14-3/14-14
TSEC1_GTX_CLK	TSEC1 transmit clock out	TSEC1	—	1	O	14-3/14-14
TSEC1_RXD[7:0]	TSEC1 receive data	TSEC1	—	8	I	14-3/14-14
TSEC1_RX_CLK	TSEC1 receive clock	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_DV	TSEC1 receive data valid	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_ER	TSEC1 receiver error	TSEC1	—	1	I	14-3/14-14
TSEC1_TXD[7:0]	TSEC1 transmit data 7–0	TSEC1	—	8	O	14-3/14-14
TSEC1_TX_CLK	TSEC1 transmit clock in	TSEC1	—	1	I	14-3/14-14
TSEC1_TX_EN	TSEC1 transmit enable	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_ER	TSEC1 transmit error	TSEC1	—	1	O	14-3/14-14
TSEC2_COL	TSEC2 collision detect	TSEC2	—	1	I	14-3/14-14
TSEC2_CRS	TSEC2 carrier sense	TSEC2	—	1	I	14-3/14-14
TSEC2_GTX_CLK	TSEC2 transmit clock out	TSEC2	—	1	O	14-3/14-14

Table 3-2. MPC8555E Alphabetical Signal Reference (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/Page
TSEC2_RXD[7:0]	TSEC2 receive data	TSEC2	—	8	I	14-3/14-14
TSEC2_RX_CLK	TSEC2 receive clock	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_DV	TSEC2 receive data Valid	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_ER	TSEC2 receiver error	TSEC2	—	1	I	14-3/14-14
TSEC2_TXD0	TSEC2 transmit data 0	TSEC2	cfg_pci2_clk	1	O	14-3/14-14
TSEC2_TXD1	TSEC2 transmit data 1	TSEC2	cfg_pci1_clk	1	O	14-3/14-14
TSEC2_TXD2	TSEC2 transmit data 2	TSEC2	cfg_tsec2	1	O	14-3/14-14
TSEC2_TXD3	TSEC2 transmit data 3	TSEC2	cfg_tsec1	1	O	14-3/14-14
TSEC2_TXD[7:4]	TSEC2 transmit data 7–4	TSEC2	—	4	O	14-3/14-14
TSEC2_TX_CLK	TSEC2 transmit clock in	TSEC2	—	1	I	14-3/14-14
TSEC2_TX_EN	TSEC2 transmit enable	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_ER	TSEC2 transmit error	TSEC2	—	1	O	14-3/14-14
$\overline{\text{UART\_CTS}}$[0:1]	DUART clear to send	Dual UART	—	2	I	12-2/12-3
$\overline{\text{UART\_RTS}}$[0:1]	DUART ready to send	Dual UART	—	2	O	12-2/12-3
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	12-2/12-3
UART_SOUT[0:1]	DUART serial data out	Dual UART	—	2	O	12-2/12-3
$\overline{\text{UDE}}$	Unconditional debug event	PIC	—	1	I	10-5/10-7

3.2 Configuration Signals Sampled at Reset

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

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of $\overline{\text{HRESET}}$. However, there is a setup and hold time for these signals relative to the rising edge of $\overline{\text{HRESET}}$, as described in the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications*. Note that the PLL configuration signals have different setup and hold time requirements than the other reset configuration signals.

The reset configuration signals are multiplexed with other functional signals. The values on these signals during reset are interpreted to be logic one or zero, regardless of whether the functional signal name is defined as active low. Most of the reset configuration signals have internal pull-up resistors so that if the signals are not driven, the default value is high (a one), as shown in the table. Some signals do not have pull-up resistors and must be driven high or low during the reset period. For details about all the signals that require external pull-up resistors, see the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications*.

Note that the multiplexing of various signals on the MPC8555E is controlled by the PMUXCR register described in Chapter 18, "Global Utilities." Also, the multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 45, "Parallel I/O Ports," for details on CPM signal multiplexing.

Table 3-3. MPC8555E Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
PCI	PCI1_GNT4	cfg_pci1_hold_en	1
	PCI1_GNT3	cfg_pci1_debug	1
	PCI1_GNT2	cfg_pci1_arb	1
	PCI1_GNT1	cfg_pci1_impd	1
PCI2	PCI1_REQ64/ PCI2_FRAME	cfg_pci1_width	1
	PCI2_GNT4	cfg_pci2_hold_en	1
	PCI2_GNT2	cfg_pci2_arb	1
	PCI2_GNT1	cfg_pci2_impd	1
Ethernet Management	EC_MDC	cfg_tsec_reduce	1
TSEC2	TSEC2_TXD3	cfg_tsec1	1
	TSEC2_TXD2	cfg_tsec2	1
	TSEC2_TXD1	cfg_pci1_clk	1
	TSEC2_TXD0	cfg_pci2_clk	1
LBC	LA27	cfg_cpu_boot	1
	LA[28:31]	cfg_sys_pll[0:3]	Must be driven
	LWE0/LBS0	cfg_lb_hold_en0	1
	LWE1/LBS1	cfg_lb_hold_en1	1
	LWE2/LBS2	cfg_host_agt	1
	LWE3/LBS3	cfg_rom_loc2	1
	LALE	cfg_core_pll0	Must be driven
	LGPL0/LSDA10	cfg_rom_loc0	1
	LGPL1/LSDWE	cfg_rom_loc1	1
	LGPL2/LOE/LSDRAS	cfg_core_pll1	Must be driven
	LGPL3/LSDCAS	cfg_boot_seq0	1
	LGPL5	cfg_boot_seq1	1
	LAD[0:31]	cfg_gpinput[0:31]	Indeterminate if not driven (no default)
Debug	MSRCID0	cfg_mem_debug	1
	MSRCID1	cfg_ddr_debug	1

3.3 Output Signal States During Reset

When a system reset is recognized ($\overline{\text{HRESET}}$ is asserted), the MPC8555E 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 MPC8555E 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 (that is, the signals that are not multiplexed with other inputs, or are not used as reset configuration signals during system reset).

Table 3-4. Output Signal States During System Reset

Interface	Signal	State During Reset
DDR memory	MDM[0:8]	High-Z
DDR memory	MBA[0:1]	High-Z
DDR memory	MA[0:14]	High-Z
DDR memory	$\overline{\text{MWE}}$	High-Z
DDR memory	$\overline{\text{MRAS}}$	High-Z
DDR memory	$\overline{\text{MCAS}}$	High-Z
DDR memory	$\overline{\text{MCS}}$[0:3]	High-Z
DDR memory	MCKE[0:1]	Driven low
DDR memory	MCK[0:5], $\overline{\text{MCK}}$[0:5]	High-Z
TSEC1	TSEC1_TXD[3:0]	Input—reset config (test only)
TSEC1	TSEC1_TX_EN	Driven low
TSEC1	TSEC1_TX_ER	High-Z
TSEC1	TSEC1_GTX_CLK	High-Z
TSEC2	TSEC2_TXD[1:0]	Input—reset config
TSEC2	TSEC2_TX_EN	Driven low
TSEC2	TSEC2_TX_ER	High-Z
TSEC2	TSEC2_GTX_CLK	High-Z
LBC	LCLK[0:2]	High-Z
LBC	$\overline{\text{LCS}}$[0:5]	High-Z
LBC	$\overline{\text{DMA\_DACK2}}$/$\overline{\text{LCS6}}$ $\overline{\text{DMA\_DDONE2}}$/$\overline{\text{LCS7}}$	High-Z
LBC	LBCTL	Input—reset config (test only)
DMA	$\overline{\text{DMA\_DACK}}$[0:1]	High-Z
DMA	$\overline{\text{DMA\_DACK2}}$/$\overline{\text{LCS6}}$ $\overline{\text{DMA\_DDONE2}}$/$\overline{\text{LCS7}}$	High-Z
DMA	$\overline{\text{DMA\_DDONE0}}$	High-Z

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

Interface	Signal	State During Reset
DMA	$\overline{\text{DMA\_DDONE1}}$	Driven (test only)
PIC	IRQ8	Driven (test only)
PIC	$\overline{\text{IRQ\_OUT}}$	High-Z
Dual UART	UART_SOUT[0:1]	High-Z
Dual UART	$\overline{\text{UART\_RTS}}$[0:1]	High-Z
System control	$\overline{\text{HRESET\_REQ}}$	High-Z
System control	$\overline{\text{CKSTP\_OUT}}$	High-Z
Debug	TRIG_OUT/READY	Input—reset config (test only)
Debug	MSRCID[2:4]	High-Z
Debug	MDVAL	High-Z
Power mgmt	ASLEEP	Input—reset config (test only)
Clock	CLK_OUT	High-Z

Chapter 4
Reset, Clocking, and Initialization

This chapter describes the reset, clocking, and some overall initialization of the MPC8555E, including a definition of the reset configurations signals and the options they select. Additionally, the configuration, control, and status registers are described. Note that chapters in this book describe specific initialization aspects for individual blocks.

4.1 Overview

The reset, clocking, and control signals provide many options for the operation of the MPC8555E. Additionally, many modes are selected with reset configuration signals during the assertion of a hard reset (assertion of $\overline{\text{HRESET}}$).

4.2 External Signal Description

Table 4-1 summarizes the external signals described in this chapter. Table 4-2 and Table 4-3 have detailed signal descriptions, but Table 4-1 contains references to additional sections that contain more information.

Table 4-1. Signal Summary

Signal	I/O	Description	References (Section/Page)
$\overline{\text{HRESET}}$	I	Hard reset input. Causes a power-on reset (POR) sequence	4.4.1.2/4-8
$\overline{\text{HRESET\_REQ}}$	O	Hard reset request output. An internal block requests that $\overline{\text{HRESET}}$ be asserted	—
$\overline{\text{SRESET}}$	I	Soft reset input. Causes *mcp* assertion to the core and a soft reset to the CPM	4.4.1.1/4-8
READY	O	The MPC8555E has completed the reset operation and is not in a power-down (nap, doze or sleep) or debug state.	4.4.2/4-9
SYSCLK	I	Primary clock input to the MPC8555E	4.4.4.1/4-21
PCI1_CLK	I	PCI clock input for PCI1 in asynchronous mode	4.4.4/4-21
PCI2_CLK	I	PCI clock input for PCI2 in asynchronous mode	4.4.4/4-21
RTC	I	Real time clock input	4.4.4.3/4-22

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

4.2.1 System Control Signals

Table 4-2 describes some of the system control signals of the MPC8555E. Section 4.4.3, “Power-On Reset Configuration,” describes the signals that also function as reset configuration signals. Note that the $\overline{\text{CKSTP\_IN}}$ and $\overline{\text{CKSTP\_OUT}}$ signals are described in Chapter 18, “Global Utilities.”

Table 4-2. System Control Signals—Detailed Signal Descriptions

Signal	I/O	Description	
HRESET	I	Hard reset. Causes the MPC8555E to abort all current internal and external transactions and set all registers to their default values. HRESET may be asserted completely asynchronously with respect to all other signals.	
		State Meaning	Asserted/Negated—See Chapter 3, "Signal Descriptions," and Section 4.4.3, "Power-On Reset Configuration," for more information on the interpretation of the other MPC8555E signals during reset.
		Timing	Assertion/Negation—The *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* gives specific timing information for this signal and the reset configuration signals.
HRESET_REQ	O	Hard reset request. Indicates to the board (system in which the MPC8555E is embedded) that a condition requiring the assertion of HRESET has been detected.	
		State Meaning	Asserted—A watchdog timer or a boot sequencer failure (see Section 11.4.5, "Boot Sequencer Mode,") has triggered a request for hard reset. Negated—Indicates no reset request.
		Timing	Assertion/Negation—May occur anytime, synchronous to the core complex bus clock. Once asserted, HRESET_REQ does not negate until HRESET is asserted.
SRESET	I	Soft reset. Causes a machine check interrupt to the e500 core and a soft reset to the CPM. Note that if the e500 core is not configured to process machine check interrupts, the assertion of SRESET causes a core checkstop. SRESET need not be asserted during a hard reset.	
		State Meaning	Asserted—Asserting SRESET causes a machine check interrupt (edge sensitive) to the e500 core and a soft reset to the CPM (level sensitive). SRESET has no effect while HRESET is asserted. However, the POR sequence is paused if SRESET is asserted during POR.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Must be asserted for at least two *CCB_clk* cycles.
READY	O	Ready. Multiplexed with TRIG_OUT. See Chapter 20, "Debug Features and Watchpoint Facility," for more information on TOSR and TRIG_OUT.	
		State Meaning	Asserted—Indicates that the MPC8555E has completed the reset operation and is not in a power-down state (nap, doze or sleep) when TOSR[SEL] equals 0b000. See Section 4.4.2, "Power-On Reset Sequence," for more information.
		Timing	Assertion/Negation—Initial assertion of READY after reset is synchronous with SYSCLK. Subsequent assertion/negation due to power down modes occurs asynchronously.

4.2.2 Clock Signals

Table 4-3 describes the overall clock signals of the MPC8555E. Note that some clock signals are specific to blocks within the MPC8555E, and although some of their functionality is described in Section 4.4.4, "Clocking," they are defined in detail in their respective chapters.

Note that there is also a CLK_OUT signal in the MPC8555E; the signal driven on the CLK_OUT pin is selectable and described in Section 18.4.1.16, "Clock Out Control Register (CLKOCR)."

Table 4-3. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description
SYSCLK	I	System clock/PCI clock (SYSCLK/PCI_CLK). SYSCLK is the primary clock input to the MPC8555E. It is the clock source for the e500 core and for all devices and interfaces that operate synchronously with the core. Multiplied up with a phased-lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock) which is used by virtually all of the synchronous system logic, include the L2 cache, the DDR SDRAM and local bus memory controllers, and other internal blocks such as the DMA and interrupt controllers. The CCB clock, in turn, feeds the PLL in the e500 core and the DLL that creates the local bus memory clocks. When the PCI interface is used in synchronous mode, SYSCLK also functions as the PCI_CLK signal. Note that this is true whether the MPC8555E is in agent or host mode. The MPC8555E does not provide a separate PCI_CLK output in host mode.
		Timing Assertion/Negation—See the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific timing information for this signal.
PCI1_CLK	I	PCI1 clock can be selected as the asynchronous PCI clock reference for the PCI1 interface of the MPC8555E. In asynchronous mode, there is no required relationship between PCI1_CLK and SYSCLK. The PCI1 interface logic can be clocked based on this input and internal logic synchronizes the PCI interface logic with the platform logic.
		Timing Assertion/Negation—See the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific timing information for this signal.
PCI2_CLK	I	PCI2 clock can be selected as the asynchronous PCI clock reference for the PCI2 interface of the MPC8555E. In asynchronous mode, there is no required relationship between PCI2_CLK and SYSCLK. The PCI2 interface logic can be clocked based on this input and internal logic synchronizes the PCI interface logic with the platform logic.
		Timing Assertion/Negation—See the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific timing information for this signal.
RTC	I	Real time clock. May be used (optionally) to clock the time base of the e500 core. The RTC timing specifications are given in the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications*, but the maximum input frequency should not exceed 1/4th the CCB frequency. See Section 4.4.4.3, “Real Time Clock.” This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).
		Timing Assertion/Negation—See the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific timing information for this signal.

4.3 Memory Map/Register Definition

There are no unique registers in the reset and clocking blocks of the MPC8555E. However, this section describes the configuration, control, and status registers of the overall MPC8555E device; it also contains a brief description of the boot sequencer.

4.3.1 Local Configuration Control

Table 4-4 shows the memory map for the configuration, control, and status registers. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 4-4. Local Configuration Control Register Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-5
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-6
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-7

4.3.1.1 Accessing Configuration, Control, and Status Registers

The configuration, control, and status registers are memory mapped. The set of configuration, control, and status registers occupies a 1-Mbyte region of memory. Their location is programmable using the CCSR base address register (CCSRBAR). The default base address for the configuration, control, and status registers is 0xFF70_0000 (CCSRBAR = 0x000F_F700). CCSRBAR itself is part of the local access block of CCSR memory, which begins at offset 0x0 from CCSRBAR. Because CCSRBAR is at offset 0x0 from the beginning of the local access registers, CCSRBAR always points to itself. The contents of CCSRBAR are broadcast internally in the MPC8555E to all functional units that need to be able to identify or create configuration transactions.

4.3.1.1.1 Updating CCSRBAR

Updates to CCSRBAR that relocate the entire 1-Mbyte region of configuration, control, and status registers, requires special treatment. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is seen. To make sure this happens, these guidelines should be followed:

- CCSRBAR should be updated during initial configuration of the device when only one host or controller has access to the device.
  - If the boot sequencer is being used to initialize, it is recommended that the boot sequencer set CCSRBAR to its desired final location.
  - If an external host on PCI is configuring the device, it should set CCSRBAR to the desired final location before the e500 core is released to boot.
  - If the e500 core is initializing the device, it should set CCSRBAR to the desired final location before enabling other I/O devices to access the device.
- When the e500 core is writing to CCSRBAR, it should use the following sequence:
  - Read the current value of CCSRBAR using a load word instruction followed by an **isync**. This forces all accesses to configuration space to complete.
  - Write the new value to CCSRBAR.
  - 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 CCSRBAR from its new location, followed by another **isync** instruction.

4.3.1.1.2 Configuration, Control, and Status Base Address Register (CCSRBAR)

Figure 4-1 shows the fields of CCSRBAR.

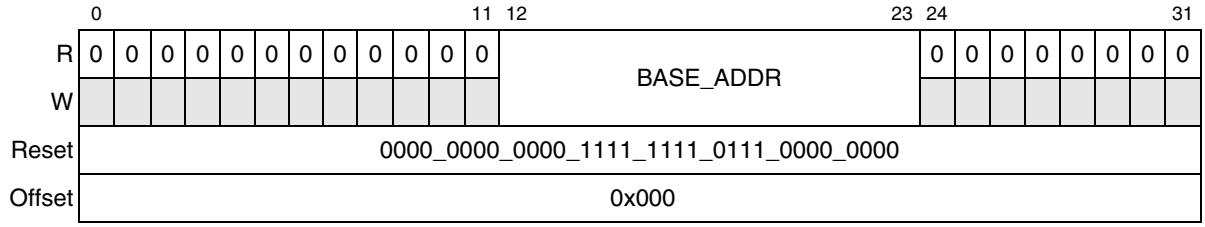


Figure 4-1. Configuration, Control, and Status Register Base Address Register (CCSRBAR)

Table 4-5 defines the bit fields of CCSRBAR.

Table 4-5. CCSRBAR Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0
12–23	BASE_ADDR	Identifies the12 most significant address bits of the window used for configuration accesses. The base address is aligned on a 1-Mbyte boundary.
24–31	—	Write reserved, read = 0

4.3.1.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBAR and ALTCAR registers. These are intended to be used with the boot sequencer to allow the boot sequencer to access an alternate 1-Mbyte region of configuration space. 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 that is mapped to the target specified in ALTCAR. Thus, by configuring these registers, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See Section 11.4.5, “Boot Sequencer Mode,” for more information.

NOTE

The enable bit in the ALTCAR register should be cleared either by the boot sequencer or by the boot code that executes after the boot sequencer has completed its configuration operations. This prevents problems with incorrect mappings if subsequent configuration of the local access windows uses a different target mapping for the address specified in ALTCBAR.

4.3.1.2.1 Alternate Configuration Base Address Register (ALTCBAR)

Figure 4-2 shows the fields of ALTCBAR.

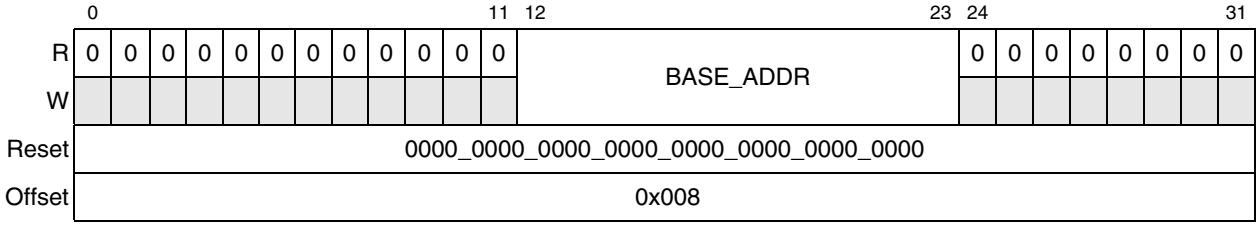


Figure 4-2. Alternate Configuration Base Address Register (ALTCBAR)

Table 4-6 defines the bit fields of ALTCBAR.

Table 4-6. ALTCBAR Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0
12–23	BASE_ADDR	Identifies the12 most significant address bits of an alternate window used for configuration accesses
24–31	—	Write reserved, read = 0

4.3.1.2.2 Alternate Configuration Attribute Register (ALTCAR)

Figure 4-3 shows the fields of ALTCAR.

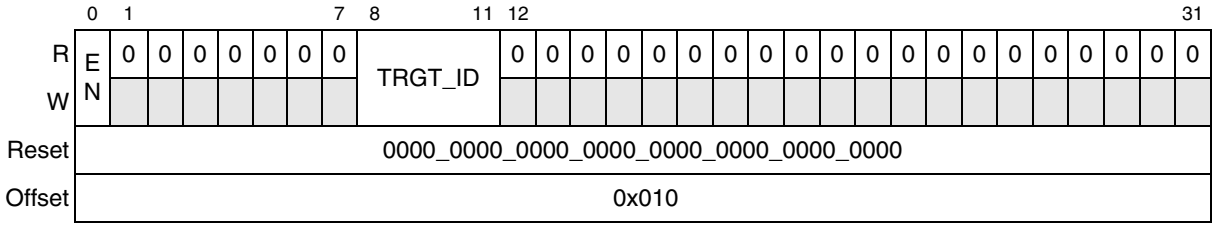


Figure 4-3. Alternate Configuration Attribute Register (ALTCAR)

Table 4-7 defines ALTCAR fields.

Table 4-7. ALTCAR Bit Settings

Bits	Name	Description
0	EN	Enable for a second configuration window. Like CCSRBAR, it has a fixed size of 1 Mbyte. 0 Second configuration window is disabled 1 Second configuration window is enabled
1–7	—	Write reserved, read = 0

Table 4-7. ALTCAR Bit Settings (continued)

Bits	Name	Description
8–11	TRGT_ID	Identifies the device ID to target when a transaction hits in the 1-Mbyte address range defined by the second configuration window. 0000 PCI interface 1 0001 PCI interface 2 0010 Reserved 0011 Reserved 0100 Local bus controller 0101–0111 Reserved 1000 Configuration, control, and status registers 1001–1110 Reserved 1111 Local memory —DDR SDRAM and on-chip SRAM
12–31	—	Write reserved, read = 0

4.3.1.3 Boot Page Translation

When the e500 core comes out of reset, its MMU has one 4-Kbyte page defined at 0xFFFF_F*nnn*. The first instruction executed by the e500 core is always address 0xFFFF_FFFC, which must be a branch to an address within the 4-Kbyte boot page. For systems in which the boot code resides at a different address, the MPC8555E provides boot page translation capability. Boot page translation is controlled by the boot page translation register (BPTR).

The boot sequencer can enable boot page translation, or the boot page can be translated by an external host when the MPC8555E is configured to be in boot holdoff mode. If translation is performed to a page outside of the default boot ROM address range defined in the MPC8555E (8 Mbytes at 0xFF80_0000 to 0xFFFF_FFFF as defined in Section 4.4.3.3, “Boot ROM Location”), the external host or boot sequencer must also set up a local access window to define the routing of the boot code fetch to the target interface that contains the boot code because the BPTR defines only the address translation, not the target interface. See Section 2.1, “Local Memory Map Overview and Example,” and Section 11.4.5, “Boot Sequencer Mode,” for more information.

4.3.1.3.1 Boot Page Translation Register (BPTR)

Figure 4-4 shows the fields of BPTR.

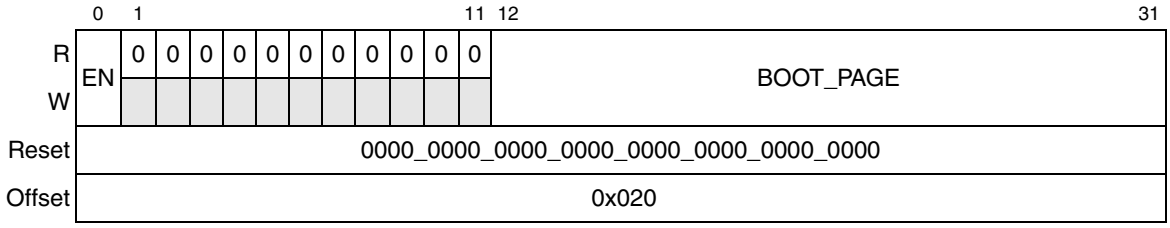


Figure 4-4. Boot Page Translation Register (BPTR)

Table 4-8 describes BPTR bit settings.

Table 4-8. BPTR Bit Settings

Bits	Name	Description
0	EN	Boot page translation enable 0 Boot page is not translated 1 Boot page is translated as defined in the BPTR[BOOT_PAGE] parameter
1–11	—	Write reserved, read = 0
12–31	BOOT_PAGE	Translation for boot page. If enabled, the high order 20 bits of accesses to 0xFFFF_F*nnn* are replaced with this value.

4.3.2 Boot Sequencer

The boot sequencer is a DMA engine that accesses a serial ROM on the I^2C interface and writes data to CCSR memory or the memory space pointed to by the alternate configuration base address register (ALTCBAR). See Section 4.3.1.2, "Accessing Alternate Configuration Space." The boot sequencer is enabled by reset configuration pins as described in Section 4.4.3.6, "Boot Sequencer Configuration." If the boot sequencer is enabled, the e500 core is held in reset until the boot sequencer has completed its operation. For more details, see Section 11.4.5, "Boot Sequencer Mode."

4.4 Functional Description

This section describes the various ways to reset the MPC8555E device, the POR configurations, and the clocking on the MPC8555E.

4.4.1 Reset Operations

The MPC8555E has reset input signals for hard and soft reset operation.

4.4.1.1 Soft Reset

Assertion of the $\overline{\text{SRESET}}$ signal causes the CPM to go through its soft reset sequence. In addition, assertion of $\overline{\text{SRESET}}$ causes a machine check interrupt to the e500 core. When this occurs, the soft reset flag is recorded in the machine check summary register (MCPSUMR) in the global utilities block so that software can identify the machine check as a soft reset condition. See the *PowerPC™ e500 Core Family Reference Manual* for more information on the machine check interrupt, and Section 18.4.1.13, "Machine Check Summary Register (MCPSUMR)," for more information on the setting of the soft reset flag. Note that if $\overline{\text{SRESET}}$ is asserted before the e500 core is configured to handle a machine check interrupt, a core checkstop condition occurs, which causes $\overline{\text{CKSTP\_OUT}}$ to assert.

4.4.1.2 Hard Reset

The MPC8555E can be completely reset by the assertion of the $\overline{\text{HRESET}}$ input. The assertion of this signal by external logic is the equivalent of a POR and causes the sequence of events described in Section 4.4.2, "Power-On Reset Sequence."

Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for the timing requirements for HRESET assertion and negation.

The MPC8555E hard reset request output signal (HRESET_REQ) indicates to external logic that a hard reset is being requested by hardware. Hardware causes this signal to assert for a boot sequencer failure (see Section 11.4.5, "Boot Sequencer Mode," and Section 11.4.5.2, "EEPROM Data Format,") or when the e500 watchdog timer is configured to cause a reset request when it expires.

4.4.2 Power-On Reset Sequence

The POR sequence for the MPC8555E is as follows:

1. Power is applied to meet the specifications in the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications*.
2. System asserts HRESET and TRST causing all registers to be initialized to their default states and most I/O drivers to be three-stated (some clock, clock enabled, and system control signals are active).
3. System applies a stable SYSCLK signal and stable PLL configuration inputs, and the device PLL begins locking to SYSCLK.
4. System negates HRESET after its required hold time and after POR configuration inputs have been valid for at least 4 SYSCLK cycles.

NOTE

If the JTAG signals are not used, then TRST may be tied active; however, it is recommended that TRST not remain asserted after the negation of HRESET. TRST may be connected directly to HRESET.

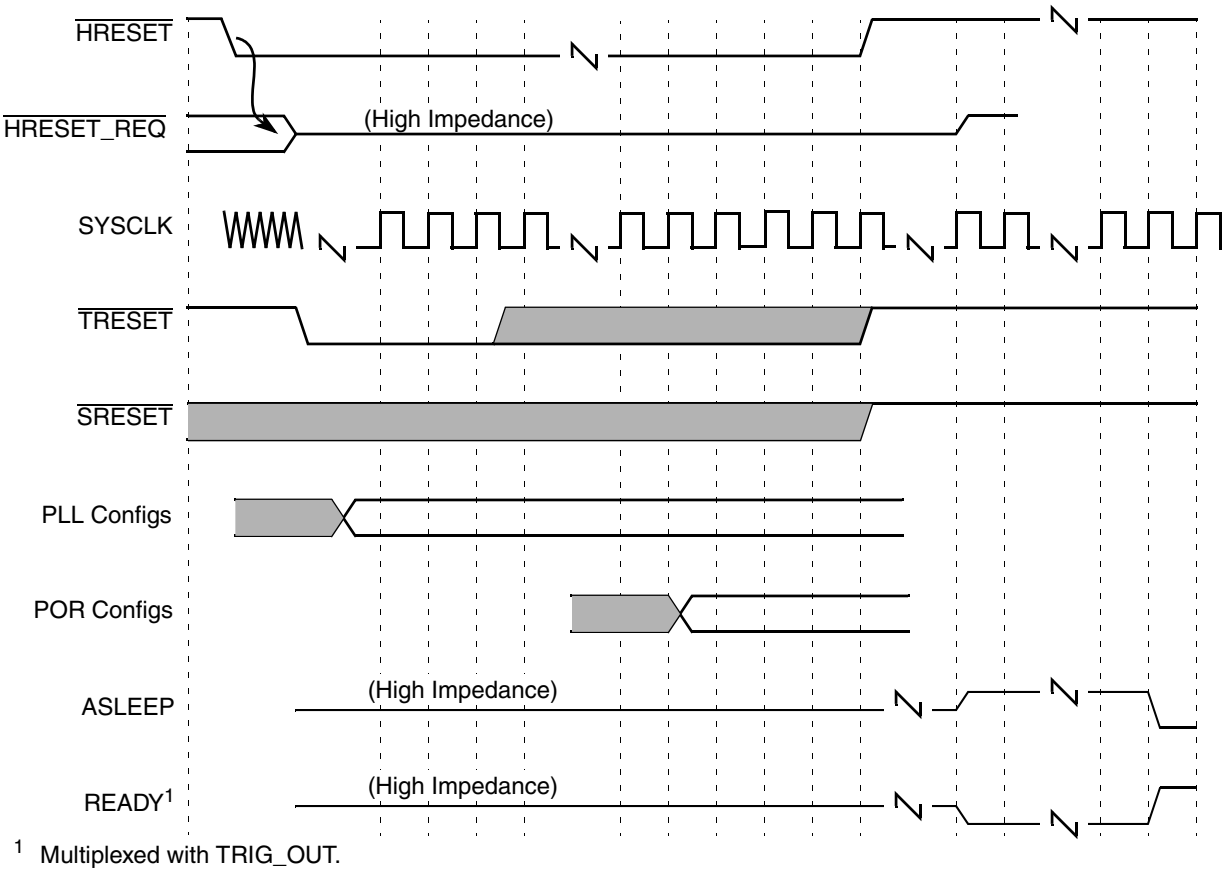
There is no need to assert the SRESET signal when HRESET is asserted. If SRESET is asserted on negation of HRESET, the POR sequence will be paused after the e500 core PLL is locked and before the e500 reset is negated. The POR sequence will be resumed when SRESET is negated.

5. MPC8555E enables I/O drivers.
6. The MPC8555E PCI interface can assert DEVSEL in response to configuration cycles.
7. The CPM reset signals (cpm_por_reset, cpm_hreset, and cpm_sreset) are asserted, and the e500 and CPM PLL configuration inputs are applied, allowing the e500 and CPM PLLs to begin locking to the device clock (the CCB clock).
8. The CCB clock is cycled for approximately 50 μs to lock the e500 PLL and the CPM PLL.
9. The internal hard reset to the e500 core is negated and soft resets are negated to the DLL and other remaining I/O blocks. The DLL begins to lock.
10. When DLL locking is completed, the boot sequencer is released, causing it to load configuration data from serial ROMs, if enabled, as described in Section 4.4.3.6, "Boot Sequencer Configuration."

11. When the boot sequencer completes, the PCI interface is released to accept external requests, and the boot vector fetch by the e500 core is allowed to proceed unless processor booting is further held off by POR configuration inputs as described in Section 4.4.3.5, “CPU Boot Configuration.” The MPC8555E is now in its ready state.
12. The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state. The ready state is also indicated by the assertion of READY/TRIG_OUT if TOSR[SEL] = 000. In this case, READY is asserted with the same rising edge of SYSCLK, to indicate that the device has reached its ready state. See Section 20.3.4.1, “Trigger Out Source Register (TOSR),” for more information on this register.

    Asserting READY allows external system monitors to know basic device status. For example, exactly when it emerges from reset, or if the device is in a low-power mode. For more information on the debug functions of TRIG_OUT, see Section 20.3.4, “Trigger Out Function.” For more information about power management states, see Section 18.4.1, “Register Descriptions.”

Figure 4-5 shows a timing diagram of the POR sequence.



1 Multiplexed with TRIG_OUT.

Figure 4-5. Power-on Reset Sequence

4.4.3 Power-On Reset Configuration

Various device functions are initialized by sampling certain signals during the assertion of $\overline{\text{HRESET}}$. The values of all these signals are sampled into registers while $\overline{\text{HRESET}}$ is asserted. These inputs are to be pulled high or low by external resistors. During $\overline{\text{HRESET}}$, all other signal drivers connected to these signals must be in the high-impedance state.

All POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board.

This section describes the functions and modes configured by POR configuration signals. Note that many reset configuration settings are accessible to software through the following read-only memory-mapped registers described in Chapter 18, "Global Utilities":

- POR PLL status register (PORPLLSR)
- POR boot mode status register (PORBMSR)
- POR I/O impedance status and control register (PORIMPSCR)
- POR device status register (PORDEVSR)
- POR debug mode status register (PORDBGMSR)
- General-purpose POR configuration register (GPPORCR)—reports the value on LAD[0:31] during POR (can be used to external system configuration)

NOTE

In the following tables, the binary value 0b0 represents a signal pulled down to GND and a value of 0b1 represents a signal pulled up to V_{DD}, regardless of the sense of the functional signal name on the signal.

4.4.3.1 System PLL Ratio

The system PLL inputs, shown in Table 4-9, establish the clock ratio between the PCI_CLK/SYSCLK input and the platform clock used by the MPC8555E. The platform clock, also called the CCB clock, drives the L2 cache, the DDR SDRAM data rate, and the e500 core complex bus (CCB). There is no default value for this PLL ratio; these signals must be pulled to the desired values. Note that the values latched on these signals during POR are accessible in the PORPLLSR (POR PLL status register), as described in Section 18.4.1.1, "POR PLL Status Register (PORPLLSR)."

Table 4-9. CCB Clock PLL Ratio

Functional Signals	Reset Configuration Name	Value (Binary)	CCB Clock : SYSCLK Ratio
LA[28:31] No default	cfg_sys_pll[0:3]	0000	16 : 1
		0001	Reserved
		0010	2 : 1
		0011	3 : 1
		0100	4 : 1
		0101	5 : 1
		0110	6 : 1
		0111	Reserved
		1000	8 : 1
		1001	9 : 1
		1010	10 : 1
		1011	Reserved
		1100	12 : 1
		1101	Reserved
		1110	Reserved
		1111	Reserved

4.4.3.2 e500 Core PLL Ratio

Table 4-10 describes the e500 core clock PLL inputs that program the core PLL and establish the ratio between the e500 core clock and the e500 core complex bus (CCB) clock. There is no default value for this PLL ratio; these signals must be pulled to the desired values. Note that the values latched on these signals during POR are accessible through the memory-mapped PORPLLSR, as described in Section 18.4.1.1, "POR PLL Status Register (PORPLLSR)," and also in the e500 core HID1 register, as described in Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."

Table 4-10. e500 Core Clock PLL Ratios

Functional Signals	Reset Configuration Name	Value (Binary)	e500 Core: CCB Clock Ratio
LALE, LGPL2 No default	cfg_core_pll[0:1]	00	2 : 1
		01	5 : 2 (2.5:1)
		10	3 : 1
		11	7 : 2 (3.5:1)

4.4.3.3 Boot ROM Location

The first instruction executed by the e500 core is always address 0xFFFF_FFFC, which must be a branch to an address within the 4-Kbyte boot page. The MPC8555E defines the default boot ROM address range

to be 8 Mbytes at address 0xFF80_0000 to 0xFFFF_FFFF. However, which on-chip peripheral handles these boot ROM accesses can be selected at power up.

The boot ROM location inputs, shown in Table 4-11, establish 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 these inputs.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 18.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-11. Boot ROM Location

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LGPL0, LGPL1, $\overline{\text{LWE}}$[3] Default (111)	cfg_rom_loc[0:2]	000	PCI 1
		001	DDR SDRAM
		010	PCI 2
		011	Reserved
		100	Reserved
		101	Local bus GPCM—8-bit ROM
		110	Local bus GPCM—16-bit ROM
		111	Local bus GPCM—32-bit ROM (default)

See Section 2.1, "Local Memory Map Overview and Example," for an example memory map that relies on the default boot ROM values. Also, see Section 4.3.1.3.1, "Boot Page Translation Register (BPTR)," for information on translation of the boot page. If enabled, this translation only affects CPU accesses to 0xFFFF_F*nnn*.

4.4.3.4 Host/Agent Configuration

The host/agent reset configuration inputs, shown in Table 4-12, configure the MPC8555E to act as a host or as an agent of a master on another interface. In host mode, the MPC8555E is immediately enabled to master transactions to the PCI interfaces. If the MPC8555E is an agent on the PCI1 interface, then the MPC8555E is disabled from mastering transactions on that interface until the external host enables it to do so. The external host does this by setting the control registers of the MPC8555E interfaces appropriately. See details in the PCI programming models described in Chapter 16, "PCI Bus Interface."

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 18.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-12. Host/Agent Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE}}$[2] Default (1)	cfg_host_agt	0	MPC8555E acts as an agent on PCI1
		1	MPC8555E acts as a host on PCI1

NOTE

The MPC8555E is always a host on PCI2. If the MPC8555E is an agent on PCI1, and the CPU is not in holdoff mode (as described in Section 4.4.3.5, "CPU Boot Configuration"), the boot ROM should not be located on PCI1 where the external host exists, because the MPC8555E is not initially enabled to master reads onto that interface.

4.4.3.5 CPU Boot Configuration

The CPU boot configuration input, shown in Table 4-13, specifies the boot configuration mode. If LA27 is sampled low at reset, the e500 core is prevented from fetching boot code until configuration by an external master is complete. The external master frees the CPU to boot by setting EEBPCR[CPU_EN] in the ECM CCB port configuration register (EEBPCR). See Section 8.2.1.2, "ECM CCB Port Configuration Register (EEBPCR)," for more information.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 18.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-13. CPU Boot Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LA27 Default (1)	cfg_cpu_boot	0	CPU boot hold-off mode. The e500 core is prevented from booting until configured by an external master.
		1	The e500 core is allowed to boot without waiting for configuration by an external master (default).

4.4.3.6 Boot Sequencer Configuration

The boot sequencer configuration options, shown in Table 4-14, allow the boot sequencer to load configuration data from the serial ROM located on the I^2C port before the host tries to configure the MPC8555E. These options also specify normal or extended I^2C addressing modes. See Section 11.4.5, "Boot Sequencer Mode," for more information on the boot sequencer.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 18.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-14. Boot Sequencer Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LGPL3, LGPL5 Default (11)	cfg_boot_seq[0:1]	00	Reserved
		01	Normal I^2C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I^2C interface. A valid ROM must be present.
		10	Extended I^2C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I^2C interface. A valid ROM must be present.
		11	Boot sequencer is disabled. No I^2C ROM is accessed (default).

NOTE

When the boot sequencer is enabled, the processor core will be held in reset and thus prevented from fetching boot code until the boot sequencer has completed its task, regardless of the state of the CPU boot configuration signal described in Section 4.4.3.5, "CPU Boot Configuration."

4.4.3.7 TSEC Width

The TSEC width input, shown in Table 4-15, selects standard versus reduced width for both of the three-speed Ethernet controller interfaces. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 18.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-15. TSEC Width Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
EC_MDC Default (1)	cfg_tsec_reduce	0	Ethernet interfaces operate in reduced mode, either RTBI or RGMII, using only four transmit data signals and four receive data signals.
		1	Ethernet interfaces operate in their standard TBI or GMII modes using eight transmit data signals and eight receive data signals (default).

NOTE

While the width of both interfaces is controlled by this single configuration input, the protocol (TBI or GMII) used by each is separately controlled with other configuration inputs described in Section 4.4.3.8, "TSEC1 Protocol," and Section 4.4.3.9, "TSEC2 Protocol."

4.4.3.8 TSEC1 Protocol

The TSEC1 protocol input, shown in Table 4-16, selects the protocol (GMII or TBI) used by the TSEC1 controller. Note that the value latched on this signal during POR is accessible through the memory-mapped

PORDEVSR (POR device status register) described in Section 18.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-16. TSEC1 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD3 Default (1)	cfg_tsec1	0	The TSEC1 controller operates using the GMII protocol (or RGMII if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width.")
		1	The TSEC1 controller operates using the TBI protocol (or RTBI if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width,") (default).

4.4.3.9 TSEC2 Protocol

The TSEC2 protocol input, shown in Table 4-17, selects the protocol (GMII or TBI) used by the TSEC2 controller. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 18.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-17. TSEC2 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD2 Default (1)	cfg_tsec2	0	The TSEC2 controller operates using the GMII protocol (or RGMII if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width.")
		1	The TSEC2 controller operates using the TBI protocol (or RTBI if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width,") (default).

4.4.3.10 PCI Clock Selection

The PCI clock source inputs, shown in Table 4-18 and Table 4-19 specify the clock mode (synchronous or asynchronous) for the PCI1 and PCI2 interfaces. See Section 4.4.4.1, "System Clock and PCI Clocks," for more information. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 18.4.1.1, "POR PLL Status Register (PORPLLSR)."

Table 4-18. PCI1 Clock Select

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD1 Default (1)	cfg_pci1_clk	0	Asynchronous mode. PCI1_CLK is used as the clock for the PCI1 interface
		1	Synchronous mode. SYSCLK is used as the clock for the PCI1 interface (default)

Table 4-19. PCI2 Clock Select

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD0 Default (1)	cfg_pci2_clk	0	Asynchronous mode. PCI2_CLK is used as the clock for the PCI2 interface
		1	Synchronous mode. SYSCLK is used as the clock for the PCI2 interface (default)

4.4.3.11 PCI Width Configuration

The PCI width configuration input, shown in Table 4-20, configures the PCI1 interface to 32- or 64-bit operation. Note that the value latched on this signal during POR is accessible through the PORDEVSR described in Section 18.4.1.4, "POR Device Status Register (PORDEVSR)." Note also that the PCI1_REQ64_B signal used for this configuration input is physically the same pin as PCI2_FRAME_B, but if the second PCI port is to be used, then there is no need to configure the width since the default 32-bit interface must be used.

Table 4-20. PCI-32 Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI2\_FRAME}}$	cfg_pci1_width	0	The PCI interface operates as a 64-bit interface.
Default (1)		1	The PCI interface operates as two 32-bit interfaces (default).

4.4.3.12 PCI I/O Impedance

The PCI I/O impedance inputs, shown in Table 4-21 and Table 4-22, select the impedance of the PCI I/O drivers. Note that the values latched on these signals during POR are accessible through PORIMPSCR, described in Section 18.4.1.3, "POR I/O Impedance Status and Control Register (PORIMPSCR)." Note that if a 64-bit PCI interface is configured, the user is responsible to ensure that both the upper and lower 32 bits possess the same I/O impedance configuration.

Table 4-21. PCI1 I/O Impedance

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI1\_GNT1}}$	cfg_pci1_impd	0	25-Ω I/O drivers are used on the PCI1 interface (applies to the lower 32 bits if 64-bit PCI is used)
Default (1)		1	42-Ω I/O drivers are used on the PCI1 interface (applies to the lower 32 bits if 64-bit PCI is used) (default)

Table 4-22. PCI2 I/O Impedance

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI2\_GNT1}}$	cfg_pci2_impd	0	25-Ω I/O drivers are used on the PCI2 interface (applies to the upper 32 bits if 64-bit PCI is used)
Default (1)		1	42-Ω I/O drivers are used on the PCI2 interface (default)

4.4.3.13 PCI Arbiter Configuration

The PCI arbiter configuration inputs, shown in Table 4-23 and Table 4-24, enable the on-chip PCI arbiters. Note that the value latched on these signals during POR is accessible through the PORDEVSR described in Section 18.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-23. PCI1 Arbiter Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI1\_GNT2}}$	cfg_pci1_arbiter	0	The on-chip PCI arbiter is disabled for the PCI1 interface. External arbitration is required.
Default (1)		1	The on-chip PCI arbiter is enabled for the PCI1 interface (default).

Table 4-24. PCI2 Arbiter Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI2\_GNT2}}$	cfg_pci2_arbiter	0	The on-chip PCI arbiter is disabled for the PCI2 interface. External arbitration is required.
Default (1)		1	The on-chip PCI arbiter is enabled for the PCI2 interface (default).

4.4.3.14 PCI Debug Configuration

The PCI debug configuration input, shown in Table 4-25, enables PCI debug mode for the PCI1 interface only. In this mode, source ID information is driven onto the highest order address bits PCI1_AD[62:58] during the bus command phase. Note that this debug function is only available on PCI1 and only when it is being used as a 64-bit PCI interface. The value latched on this signal during POR is accessible through the PORDBGMSR described in Section 18.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-25. PCI Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI1\_GNT3}}$ Default (1)	cfg_pci_debug	0	PCI debug is enabled. Source ID information is driven onto the highest order address bits, PCI1_AD[62:58], during the bus command phase.
		1	PCI operates in normal mode (default).

4.4.3.15 Memory Debug Configuration

The memory debug configuration input, shown in Table 4-26, selects which debug outputs (DDR or LBC memory controller) are driven onto the MSRCID and MDVAL debug signals. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 18.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-26. Memory Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
MSRCID0 Default (1)	cfg_mem_debug	0	Debug information from the local bus controller (LBC) is driven on the MSRCID and MDVAL signals.
		1	Debug information from the DDR SDRAM controller is driven on the MSRCID and MDVAL signals (default).

4.4.3.16 DDR Debug Configuration

The DDR debug configuration input, shown in Table 4-27, enables a DDR memory controller debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto the 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 MPC8555E in this mode. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 18.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-27. DDR Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
MSRCID1 Default (1)	cfg_ddr_debug	0	Debug information is driven on the ECC pins instead of normal ECC I/O. ECC signals from memory devices must be disconnected.
		1	Debug information is not driven on ECC pins. ECC pins function in their normal mode (default).

4.4.3.17 PCI Output Hold Configuration

The PCI output hold configuration inputs configure the output hold times for the PCI output drivers. The default value will meet the hold times required by the specification. Hold times are adjusted for heavily loaded buses by removing the buffer delays. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific timing information. Table 4-28 shows the hold time configuration for the PCI1 interface, and Table 4-29 shows the hold time configuration for the PCI2 interface. Note that if a 64-bit PCI interface is configured, the user is responsible to ensure that both the upper and lower 32-bits possess the same output hold configuration.

Table 4-28. PCI1 Output Hold Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI1\_GNT4}}$ Default (1)	cfg_pci1_hold_en	1	Two added buffer delays—required to meet 2-ns hold time requirement. (Applies to the lower 32 bits if 64-bit PCI is used.)
		0	Zero added buffer delays—for heavily loaded systems. (Applies to the lower 32 bits if 64-bit PCI is used.)

Table 4-29. PCI2 Output Hold Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI2\_GNT4}}$ Default (1)	cfg_pci2_hold_en	1	Two added buffer delays—required to meet 2-ns hold time requirement. (Applies to upper 32 bits if 64-bit PCI is used.)
		0	Zero added buffer delays—for heavily loaded systems. (Applies to upper 32 bits if 64-bit PCI is used.)

4.4.3.18 Local Bus Output Hold Configuration

The LBC output hold configuration inputs, shown in Table 4-30, configure the output hold times for the local bus interface output drivers. Hold times are adjusted by adding buffer delays to the intrinsic delay of the output driver. The default values were selected for historical purposes. It is expected that designs done with the MPC8555E can use zero added buffer delays. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific timing information.

Table 4-30. Local Bus Output Hold Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE}}$[0:1] Default (11)	cfg_lb_hold[0:1]	11	One added buffer delay (default) (zero added buffer delays for LALE)
		10	Two added buffer delays (default + 1) (one added buffer delay for LALE)
		01	Three added buffer delays (default + 2) (one added buffer delay for LALE)
		00	Zero added buffer delays (zero added buffer delays for LALE)

4.4.3.19 General-Purpose POR Configuration

The LBC address/data bus inputs, shown in Table 4-31, configure the value of the general-purpose POR configuration register defined in Section 18.4.1.6, "General-Purpose POR Configuration Register (GPPORCR)." This register is intended to facilitate POR configuration of user systems. A value placed on LAD[0:31] during POR is captured and stored (read only) in the GPPORCR. Software can then use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

Table 4-31. General-Purpose POR Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LAD[0:31] No default	cfg_gpporcr	xx	General-purpose POR configuration vector to be placed in GPPORCR

4.4.4 Clocking

The following paragraphs describe the clocking within the MPC8555E device.

4.4.4.1 System Clock and PCI Clocks

The MPC8555E takes uses the SYSCLK input as the clock source for the e500 core and all of the devices and interfaces that operate synchronously with the core. As shown in Figure 4-6, the SYSCLK input (frequency) is multiplied up using a phase lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock). The CCB clock is used by virtually all of the synchronous system logic, including the L2 cache, and other internal blocks such as the DMA and interrupt controller. The CCB clock also feeds the PLL in the e500 core and the DLL that creates clocks for the local bus memory controller.

The PCI interfaces may be run such that the SYSCLK is the PCI clock. This is the default configuration and is called synchronous mode. However each PCI interface can be configured to use a separate PCI clock input that may be completely unrelated to the SYSCLK input; this is called asynchronous mode. Note that the divide-by-two CCB clock divider and the divide-by-*n* CCB clock divider, shown in Figure 4-6, are located in the DDR and local bus blocks, respectively.

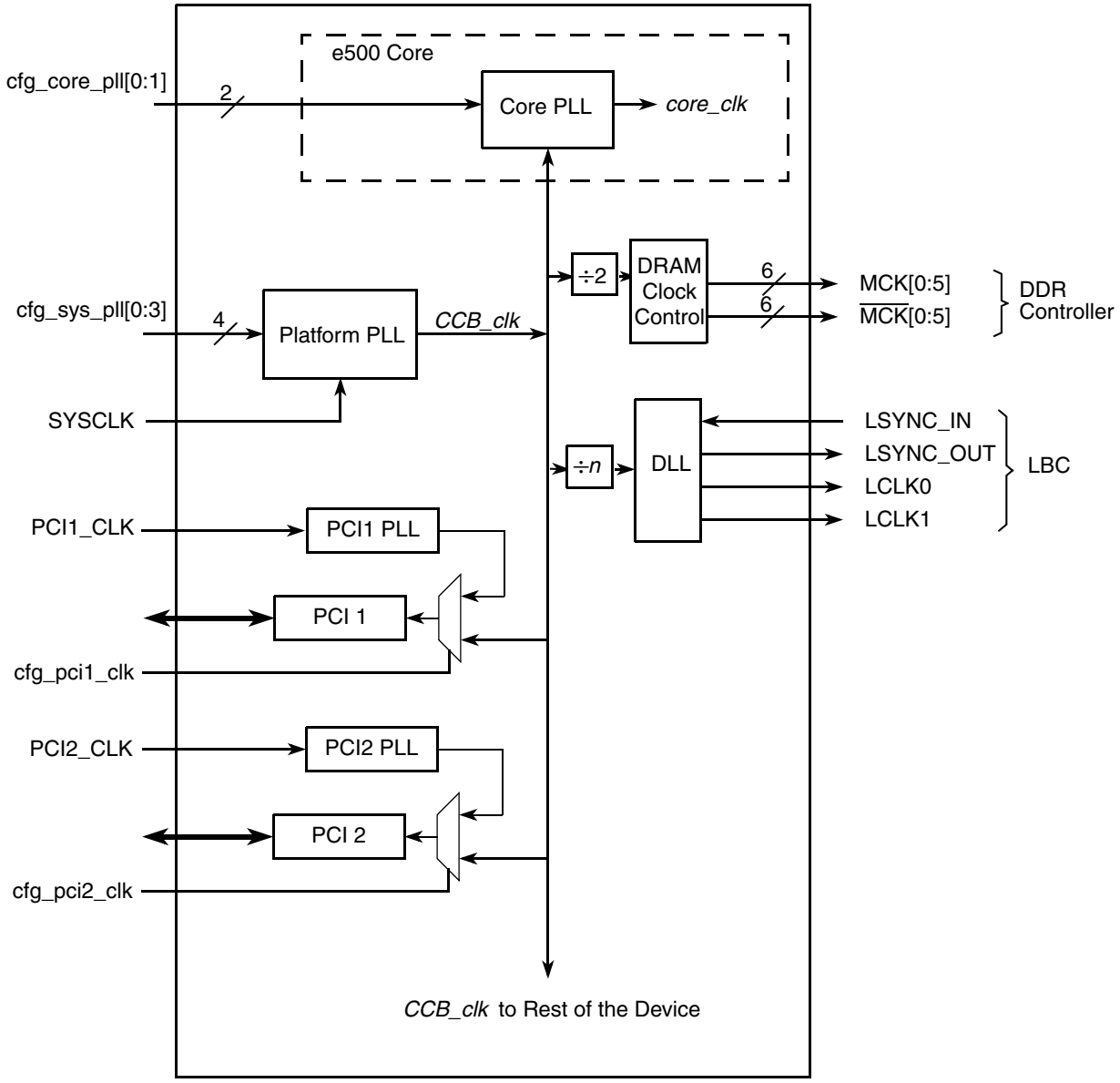


Figure 4-6. MPC8555E Clock Subsystem Block Diagram

4.4.4.2 Ethernet Clocks

The Ethernet blocks operate asynchronously with respect to the rest of the device. These blocks use receive and transmit clocks supplied by their respective PHY chips, plus a 125-MHz clock input (EC_GTX_CLK125) for gigabit protocols. Data transfers are synchronized to the CCB clock internally.

4.4.4.3 Real Time Clock

As shown in Figure 4-7, the real time clock (RTC) input can optionally be used to clock the e500 core timer facilities. RTC can also be used (optionally) by the MPC8555E programmable interrupt controller (PIC)

global timer facilities. The RTC is separate from the e500 core clock and is intended to support relatively low frequency timing applications. The RTC frequency range is specified in the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications*, but the maximum value should not exceed 1/4th of the CCB frequency.

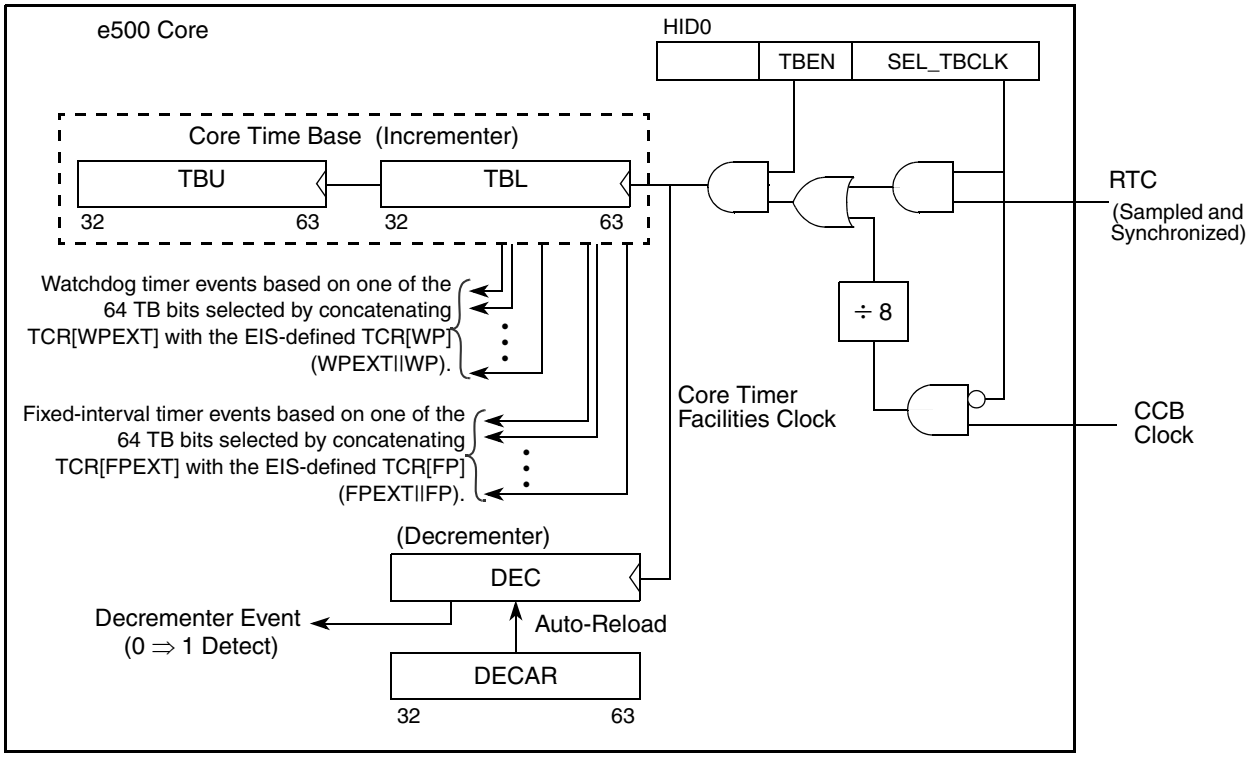
Before being distributed to the core time base, RTC is sampled and synchronized with the CCB clock.

The clock source for the core time base is specified by two fields in HID0: time base enable (TBEN), and select time base clock (SEL_TBCLK). If the time base is enabled, (HID0[TBEN] is set), the clock source is determined as follows:

- HID0[SEL_TBCLK] = 0, the time base is updated every 8 CCB clocks
- HID0[SEL_TBCLK] = 1, the time base is updated on the rising edge of RTC

The default source of the time base is the CCB clock divided by eight. For more details, see the *PowerPC™ e500 Core Family Reference Manual.*

Section 10.3.2.6, "Timer Control Register (TCR)," provides additional information on the use of the RTC signal to clock the global timers in the PIC unit.



Note: The logic circuits shown depict functional relationships only; they do not represent physical implementation details.

Figure 4-7. RTC and Core Timer Facilities Clocking Options

Part II
e500 Core Complex and L2 Cache

This part describes the many features of the MPC8555E core processor at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, “Core Complex Overview,” provides an overview of the e500 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, “Core Register Summary,” provides a listing of the e500 registers in reference form.
- Chapter 7, “L2 Look-Aside Cache/SRAM,” describes the L2 cache of the MPC8555E. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

The e500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the embedded category features of the Power Architecture technology. This part provides additional information about the architecture as it relates specifically to the e500 core complex and specific details on how its registers are accessed.

The e500 core complex interacts with the L2 cache through the core complex bus (CCB).

Chapter 5
Core Complex Overview

This chapter provides an overview of the e500 microprocessor core as it is implemented on the MPC8555E.

This chapter includes the following:

- An overview of architecture features as implemented in this core and a summary of the core feature set
- A summary of the instruction pipeline and flow
- An overview of the programming model
- An overview of interrupts and exception handling
- A description of the memory management architecture
- High-level details of the e500 core memory and coherency model
- A brief description of the core complex bus (CCB)
- A summary of the Power Architecture embedded category compatibility and migration from the original version of the PowerPC architecture as it is defined by Apple, IBM, and Motorola (referred to as the AIM version of the PowerPC architecture)

Specific details about the e500 are provided in the *PowerPC™ e500 Core Family Reference Manual* (Freescale Document ID No. E500CORERM). The e500 core provides features that the integrated device may not implement or may implement in a more specific way.

5.1 Overview

The e500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the embedded category features of the Power Architecture technology. The e500 is a 32-bit implementation using the lower words in the 64-bit general-purpose registers (GPRs).

Figure 5-1 is a block diagram of the processor core complex that shows how the functional units operate independently and in parallel. Note that this conceptual diagram does not attempt to show how these features are physically implemented.

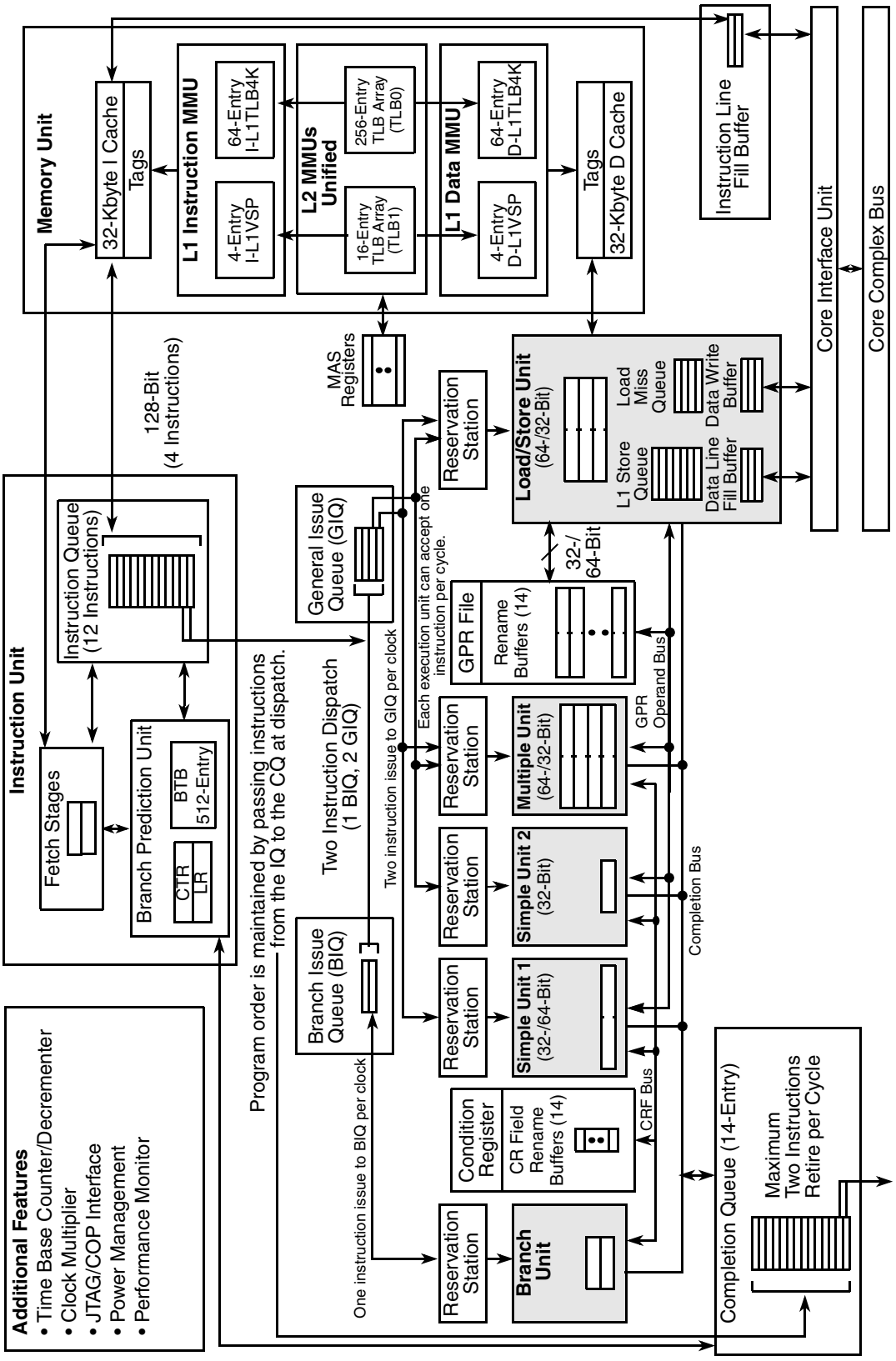


Figure 5-1. e500 Core Complex Block Diagram

The Power Architecture technology defines categories that extend the architecture that can perform computational or system management functions. One of these on the e500 is the signal processing engine (SPE), which includes a suite of vector instructions that use the upper and lower halves of the GPRs as a single two-element operand. Some extensions are defined by Freescale's embedded category implementation standards (EIS).

5.1.1 Upward Compatibility

The e500 provides 32-bit effective addresses and integer data types of 8, 16, and 32 bits, as defined by the architecture. It also provides two-element, 64-bit data types for the SPE and embedded vector floating-point instructions, which include instructions that operate on operands comprised of two 32-bit elements.

The embedded single-precision scalar floating-point instructions use 32-bit single-precision instructions.

NOTE

The SPE (which includes embedded floating-point functionality) is implemented in all PowerQUICC III devices. However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or embedded floating-point instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale Semiconductor offers a libcfsl_e500 library that uses SPE instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

5.1.2 Core Complex Summary

The core complex is a superscalar processor that can issue two instructions and complete two instructions per clock cycle. Instructions complete in order, but can execute out of order. Execution results are available to subsequent instructions through the rename buffers, but those results are recorded into architected registers in program order, maintaining a precise exception model. All arithmetic instructions that execute in the core operate on data in the GPRs. Although the GPRs are 64 bits wide, only SPE, DPFP (e500v2 only), and embedded vector floating-point instructions operate on the upper word of the GPRs; the upper 32 bits are not affected by other 32-bit instructions.

The processor core integrates two simple instruction units (SU1, SU2), a multiple-cycle instruction unit (MU), a branch unit (BU), and a load/store unit (LSU).

The LSU and SU2 support 64- and 32-bit instructions.

The ability to execute five instructions in parallel and the use of simple instructions with short execution times yield high efficiency and throughput. Most integer instructions execute in 1 clock cycle. A series of independent vector floating-point add instructions can be issued and completed with a throughput of one instruction per cycle.

The core complex includes independent on-chip, 32-Kbyte, eight-way set-associative, physically addressed caches for instructions and data. It also includes on-chip first-level instruction and data memory management units (MMUs) and an on-chip second-level unified MMU.

- The first-level MMUs contain two four-entry, fully-associative instruction and data translation lookaside buffer (TLB) arrays that provide support for demand-paged virtual memory address translation and variable-sized pages. They also contain two 64-entry, 4-way set-associative instruction and data TLB arrays that support 4-Kbyte pages. These arrays are maintained entirely by the hardware with a true least-recently-used (LRU) algorithm.

  The second-level MMU contains a 16-entry, fully-associative unified (instruction and data) TLB array that provides support for variable-sized pages. It also contains a unified TLB for 4-Kbyte page size support

The core complex allows cache-line-based user-mode locks on the contents in either the instruction or data cache. This provides embedded applications with the capability for locking interrupt routines or other important (time-sensitive) instruction sequences into the instruction cache. It also allows data to be locked into the data cache, which supports deterministic execution time.

The core complex supports a high-speed on-chip internal bus with data tagging called the core complex bus (CCB). The CCB has two general-purpose read data buses, one write data bus, data parity bits, data tag bits, an address bus, and address attribute bits. The processor core complex supports out-of-order reads, in-order writes, and one level of pipelining for addresses with address-retry responses. It can also support single-beat and burst data transfers for memory accesses and memory-mapped I/O operations.

5.2 e500 Processor and System Version Numbers

Table 5-1 lists the revision codes in the processor version register (PVR) and the system version register (SVR). These registers can be accessed as SPRs through the e500 core (see Section 6.5.3, “Processor Version Register (PVR),” and Section 6.5.4, “System Version Register (SVR)”) or as memory-mapped registers defined by the integrated device (see “Section 18.4.1.14, “Processor Version Register (PVR),” and Section 18.4.1.15, “System Version Register (SVR)”).

Table 5-1. Device Revision Level Cross-Reference

MPC8555E/ MPC8541E Revision	Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
1.1	2.0	0x8020_0020	0x8079-0011 (MPC8555E) 0x8071-0011 (MPC8555) 0x807A-0011 (MPC8541E) 0x8072-0011 (MPC8541)

5.3 Features

Key features of the e500 are summarized as follows:

- 32-bit architecture
- Additional categories (formerly referred to as APUs)

  Branch target buffer (BTB) locking is specific to the e500. BTB locking gives the user the ability to lock, unlock, and invalidate BTB entries; further information is provided in Table 5-5. The EIS (see *EREF: a Reference for Freescale Book E and the e500 Core*) defines the following:

  — Integer select. This instruction is now part of the Power Architecture technology base category.

  — Performance monitor. The performance monitor facility provides the ability to monitor and count predefined events such as processor clocks, misses in the instruction cache or data cache, types of instructions decoded, or mispredicted branches. The count of such events can be used to trigger the performance monitor exception. Additional performance monitor registers (PMRs) similar to SPRs are used to configure and track performance monitor operations. These registers are accessed with the Move to PMR and Move from PMR instructions (**mtpmr** and **mfpmr**). See Section 5.12, "Performance Monitoring."

  — Cache locking. Allows instructions and data to be locked into their respective caches on a cache block basis. Locking is performed by a set of touch and lock set instructions. This functionality can be enabled for user mode by setting MSR[UCLE]. The feature also provides resources for detecting and handling overlocking conditions.

  — Machine check. The machine check interrupt is treated as a separate level of interrupt. It uses its own save and restore registers (MCSRR0 and MCSRR1) and Return from Machine Check Interrupt (**rfmci**) instruction. See Section 5.8, "Interrupts and Exception Handling."

  — Single-precision embedded scalar and vector floating-point instructions, listed in Table 5-4.

  — Signal processing engine (SPE). Note that the SPE is not a separate unit; SPE computational and logical instructions are executed in the simple and multiple-cycle units used by all other computational and logical instructions, and 64-bit loads and stores are executed in the common LSU. Figure 5-1 shows how execution logic for SU1, the MU, and the LSU is replicated to support operations on the upper halves of the GPRs.

    The e500 register set is modified as follows:

    – GPRs are widened to 64 bits to support 64-bit load, store, and merge operations. Note that the upper 32 bits are affected only by 64-bit instructions.

    – A 64-bit accumulator (ACC) has been added.

    – The signal processing and embedded floating-point status and control register (SPEFSCR) provides interrupt control and status for SPE and embedded floating-point instructions.

    These registers are shown in Figure 5-6. SPE instructions are grouped as follows:

    – Single-cycle integer add and subtract with the same latencies for SPE operations as for the 32-bit equivalent

    – Single-cycle logical operations

    – Single-cycle shift and rotates

    – Four-cycle integer pipelined multiplies

  - 4-, 11-, 19-, and 35-cycle integer divides
  - If **r**A or **r**B is zero, a floating-point divide takes 4 cycles; all other cases take 29 cycles.
  - Four-cycle SIMD pipelined multiply-accumulate (MAC)
  - 64-bit accumulator for no-stall MAC operations
  - 64-bit loads and stores
  - 64-bit merge instructions
- Cache structure—Separate 32-Kbyte, 32-byte line, 8-way set-associative level 1 instruction and data caches
  - 1.5-cycle cache array access, 3-cycle load-to-use latency
  - Pseudo-LRU (PLRU) replacement algorithm
  - Copy-back data cache that can function as a write-through cache on a page-by-page basis
  - Supports all embedded category memory coherency modes
  - Supports EIS-defined cache-locking instructions, as listed in Table 5-3
- Dual-issue superscalar control
  - Two-instructions-per-clock peak issue rate
  - Precise exception handling
- Decode unit
  - 12-entry instruction queue (IQ)
  - Full hardware detection of interlocks
  - Decodes as many as two instructions per cycle
  - Decode serialization control
  - Register dependency resolution and renaming
- Branch prediction unit (BPU)
  - Dynamic branch prediction using a 512-entry, 4-way set-associative branch target buffer (BTB) supported by the e500 BTB instructions listed in Table 5-5.
  - Branch prediction is handled in the fetch stages.
- Completion unit
  - As many as 14 instructions allowed in 14-entry completion queue (CQ)
  - In-order retirement of as many as two instructions per cycle
  - Completion and refetch serialization control
  - Synchronization for all instruction flow changes—interrupts, mispredicted branches, and context-synchronizing instructions
- Issue queues
  - Two-entry branch instruction issue queue (BIQ)
  - Four-entry general instruction issue queue (GIQ)
- Branch unit—The branch unit (BU) is an execution unit and is distinct from the BPU. It executes (resolves) all branch and CR logical instructions.

- Two simple units (SU1 and SU2)
  - — Add and subtract
  - — Shift and rotate
  - — Logical operations
  - — Support for 64-bit SPE instructions in SU1
- Multiple-cycle unit (MU)—The MU is shown in Figure 5-2.

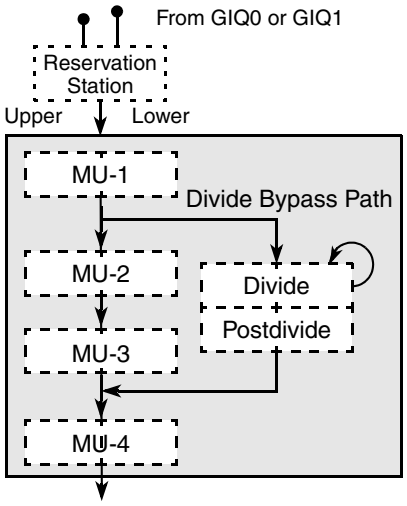


Figure 5-2. Four-Stage MU Pipeline, Showing Divide Bypass

The MU has the following features:

- — Four-cycle latency for all multiplication, including SPE integer and fractional multiply instructions and embedded scalar and vector floating-point multiply instructions
- — Variable-latency divide: 4, 11, 19, and 35 cycles for all integer divide instructions. If **r**A or **r**B is zero, floating-point divide instructions take 4 cycles; all others take 29. Note that although most divide instructions take more than 4 cycles to execute, the MU allows subsequent multiply instructions to execute through all four MU stages in parallel with the divide.
- — 4-cycle floating-point add and subtract

- The load/store unit (LSU) is shown in Figure 5-3.

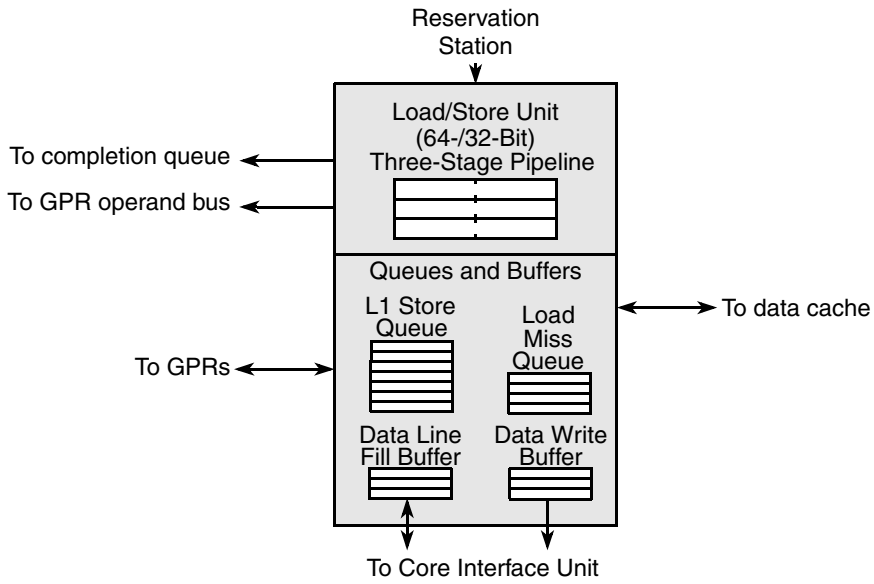


Figure 5-3. Three-Stage Load/Store Unit

The LSU has the following features:

— Three-cycle load latency

— Fully pipelined

— Load miss queue allows up to four load misses before stalling.

— Load hits can continue to be serviced when the load miss queue is full.

— The seven-entry L1 store queue allows full pipelining of stores.

— The three-entry data line fill buffer is used for loads and cacheable stores. Stores are allocated here so loads can access data from the store immediately.

— The data write buffer contains three entries: one dedicated for snoop pushes, one dedicated for castouts, and one that can be used for snoop pushes or cast outs.

- Cache coherency
  - — Supports four-state cache coherency: modified-exclusive, exclusive, shared, and invalid (MESI). Note, however that shared state may not be accessible in some implementations.
  - — Bus support for hardware-enforced coherency (bus snooping)
- Core complex bus (CCB)—internal bus
  - — High-speed, on-chip local bus with data tagging
  - — 32-bit address bus
  - — Address protocol with address pipelining and retry/copyback derived from bus used by previous generations of processors (referred to as the 60x bus)
  - — Two general-purpose read data buses and one write data bus
- Extended exception handling
  - — Supports embedded category interrupt model
    - – Less than 10-cycle interrupt latency

  - Interrupt vector prefix register (IVPR)
  - Interrupt vector offset registers (IVORs) 0–15 and 32–35
  - Exception syndrome register (ESR)
  - Preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rfci** instruction
  - A separate set of resources for machine-check interrupts
  - SPE unavailable exception
  - Floating-point data exception
  - Floating-point round exception
  - Performance monitor

- Memory management unit (MMU)
  - 32-bit effective address translated to 32-bit real address (using a 41-bit interim virtual address)
  - TLB entries for variable- (4-Kbyte–256-Mbyte) and fixed-size (4-Kbyte) pages
  - Data L1 MMU
    - 4-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Instruction L1 MMU
    - 4-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Unified L2 MMU
    - 16-entry, fully-associative TLB array for variable-sized pages
    - 256-entry, 2-way set-associative unified (for instruction and data accesses) L2 TLB array (TLB0) supports only 4-Kbyte pages
  - Software reload for TLBs
  - Virtual memory support for as much as 4 Gbytes (2^{32}) of effective address space
  - Real memory support for as much as 4 Gbytes (2^{32}) of physical memory
  - Support for big-endian and true little-endian memory on a per-page basis
- Power management
  - Low-power design
  - Power-saving modes: core-halted and core-stopped
  - Internal clock multipliers ranging from 1 to 8 times the bus clock, including integer and half-mode multipliers.The MPC8555E supports multipliers of 2, 2.5, 3, and 3.5.
  - Dynamic power management of execution units, caches, and MMUs
  - NAP, DOZE, and SLEEP bits in HID0 can be used to assert *nap*, *doze*, and *sleep* output signals to initiate power-saving modes at the integrated device level.

- Testability
  - LSSD scan design
  - JTAG interface
  - ESP support
- Reliability and serviceability
  - Parity checking on caches
  - Parity checking on e500 local bus

5.4 Instruction Set

The e500 implements the following instructions:

- The embedded category instruction set for 32-bit implementations. This is composed primarily of the user-level instructions defined by the Power Architecture user instruction set architecture (UISA). The e500 does not include floating-point instructions that require floating-point registers (FPRs), load string, or store string instructions.
- The e500 supports the following instructions:
  - Integer select. Now part of the base category. Consists of the Integer Select instruction (**isel**), which functions as an if-then-else statement that selects between two source registers by comparison to a CR bit. This instruction eliminates conditional branches, decreases latency, and reduces the code footprint.
  - Performance monitor. Table 5-2 lists performance monitor instructions.

Table 5-2. Performance Monitor Instructions

Name	Mnemonic	Syntax
Move from Performance Monitor Register	**mfpmr**	**r**D,PMRN
Move to Performance Monitor Register	**mtpmr**	PMRN,**r**S

  - Cache locking. Consists of the instructions described in Table 5-3.

Table 5-3. Cache Locking Instructions

Name	Mnemonic	Syntax
Data Cache Block Lock Clear	**dcblc**	CT, **r**A, **r**B
Data Cache Block Touch and Lock Set	**dcbtls**	CT, **r**A, **r**B
Data Cache Block Touch for Store and Lock Set	**dcbtstls**	CT, **r**A, **r**B
Instruction Cache Block Lock Clear	**icblc**	CT, **r**A, **r**B
Instruction Cache Block Touch and Lock Set	**icbtls**	CT, **r**A, **r**B

  - Machine check. Defines the Return from Machine Check Interrupt instruction (**rfmci**).
  - SPE vector instructions. Vector instructions are defined that view the 64-bit GPRs as composed of a vector of two 32-bit elements (some instructions also read or write 16-bit elements). Some scalar instructions produce a 64-bit scalar result.

— The embedded floating-point categories provide scalar and vector floating-point instructions. Scalar single-precision floating-point instructions use only the lower 32 bits of the GPRs; double-precision operands (e500v2 only) use all 64 bits. Table 5-4 lists embedded floating-point instructions.

Table 5-4. Scalar and Vector Embedded Floating-Point Instructions

Instruction	Mnemonic		Syntax
	Scalar	Vector	
Convert Floating-Point from Signed Fraction	**efscfsf**	**evfscfsf**	**r**D,**r**B
Convert Floating-Point from Signed Integer	**efscfsi**	**evfscfsi**	**r**D,**r**B
Convert Floating-Point from Unsigned Fraction	**efscfuf**	**evfscfuf**	**r**D,**r**B
Convert Floating-Point from Unsigned Integer	**efscfui**	**evfscfui**	**r**D,**r**B
Convert Floating-Point to Signed Fraction	**efsctsf**	**evfsctsf**	**r**D,**r**B
Convert Floating-Point to Signed Integer	**efsctsi**	**evfsctsi**	**r**D,**r**B
Convert Floating-Point to Signed Integer with Round toward Zero	**efsctsiz**	**evfsctsiz**	**r**D,**r**B
Convert Floating-Point to Unsigned Fraction	**efsctuf**	**evfsctuf**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer	**efsctui**	**evfsctui**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer with Round toward Zero	**efsctuiz**	**evfsctuiz**	**r**D,**r**B
Floating-Point Absolute Value	**efsabs**	**evfsabs**	**r**D,**r**A
Floating-Point Add	**efsadd**	**evfsadd**	**r**D,**r**A,**r**B
Floating-Point Compare Equal	**efscmpeq**	**evfscmpeq**	**cr**D,**r**A,**r**B
Floating-Point Compare Greater Than	**efscmpgt**	**evfscmpgt**	**cr**D,**r**A,**r**B
Floating-Point Compare Less Than	**efscmplt**	**evfscmplt**	**cr**D,**r**A,**r**B
Floating-Point Divide	**efsdiv**	**evfsdiv**	**r**D,**r**A,**r**B
Floating-Point Multiply	**efsmul**	**evfsmul**	**r**D,**r**A,**r**B
Floating-Point Negate	**efsneg**	**evfsneg**	**r**D,**r**A
Floating-Point Negative Absolute Value	**efsnabs**	**evfsnabs**	**r**D,**r**A
Floating-Point Subtract	**efssub**	**evfssub**	**r**D,**r**A,**r**B
Floating-Point Test Equal	**efststeq**	**evfststeq**	**cr**D,**r**A,**r**B
Floating-Point Test Greater Than	**efststgt**	**evfststgt**	**cr**D,**r**A,**r**B
Floating-Point Test Less Than	**efststlt**	**evfststlt**	**cr**D,**r**A,**r**B

— BTB locking instructions. The core complex provides a 512-entry BTB for efficient processing of branch instructions. The BTB is a branch target address cache, organized as 128 rows with 4-way set associativity, that holds the address and target instruction of the 512 most-recently taken branches. Table 5-5 lists BTB instructions.

Table 5-5. BTB Locking Instructions

Name	Mnemonic	Syntax
Branch Buffer Load Entry and Lock Set	**bblels**	—
Branch Buffer Entry Lock Reset	**bbelr**	—

5.5 Instruction Flow

The e500 core is a pipelined, superscalar processor with parallel execution units that allow instructions to execute out of order but record their results in order. Pipelining breaks instruction processing into discrete stages, so multiple instructions in an instruction sequence can occupy the successive stages: as an instruction completes one stage, it passes to the next, leaving the previous stage available to a subsequent instruction. So, even though it may take multiple cycles for an instruction to pass through all of the pipeline stages, once a pipeline is full, instruction throughput is much shorter than the latency.

A superscalar processor is one that issues multiple independent instructions into separate execution units, allowing parallel execution. The e500 core has five execution units, one each for branch (BU), load/store (LSU), and multiple-cycle operations (MU), and two for simple arithmetic operations (SU1 and SU2). The MU and SU1 arithmetic execution units also execute 64-bit SPE vector instructions, using both the lower and upper halves of the 64-bit GPRs.

The parallel execution units allow multiple instructions to execute in parallel and out of order. For example, a low-latency addition instruction that is issued to an SU after an integer divide is issued to the MU should finish executing before the higher latency divide instruction. The add instruction can make its results available to a subsequent instruction, but it cannot update the architected GPR specified as its target operand ahead of the multiple-cycle divide instruction.

5.5.1 Initial Instruction Fetch

The e500 core begins execution at fixed virtual address 0xFFFF_FFFC. The MMU has a default page translation which maps this to the identical physical address. So, the instruction at physical address 0xFFFF_FFFC must be a branch to another address within the 4-Kbyte boot page.

5.5.2 Branch Detection and Prediction

To improve branch performance, the e500 provides implementation-specific dynamic branch prediction using the BTB to resolve branch instructions and improve the accuracy of branch predictions. Each of the 512 entries in the 4-way set associative address cache of branch target addresses includes a 2-bit saturating branch history counter, whose value is incremented or decremented depending on whether the branch was taken. These bits can take on four values indicating strongly taken, weakly taken, weakly not taken, and strongly not taken. The BTB is used not only to predict branches, but to detect branches during the fetch stage, offering an efficient way to access instruction streams for branches predicted as taken.

In the e500, all branch instructions are assigned positions in the completion queue at dispatch. Speculative instructions in branch target streams are allowed to execute and proceed through the completion queue, although they can complete only after the branch prediction is resolved as correct and after the branch instruction itself completes.

If a branch resolves as correct, instructions in the target stream are marked nonspeculative and are allowed to complete. If the branch history bits in the BTB indicated weakly taken or weakly not taken, the prediction is upgraded to strongly taken or strongly not taken.

If a branch resolves as incorrect, instructions in the target stream are flushed from the execution pipeline, the branch history bits are updated in the BTB entry, and nonspeculative fetching begins from the correct path.

5.5.3 e500 Execution Pipeline

The seven stages of the e500 execution pipeline—fetch1, fetch2/predecode, decode/dispatch, issue, execute, complete, and write back—are highlighted in grey in Figure 5-4.

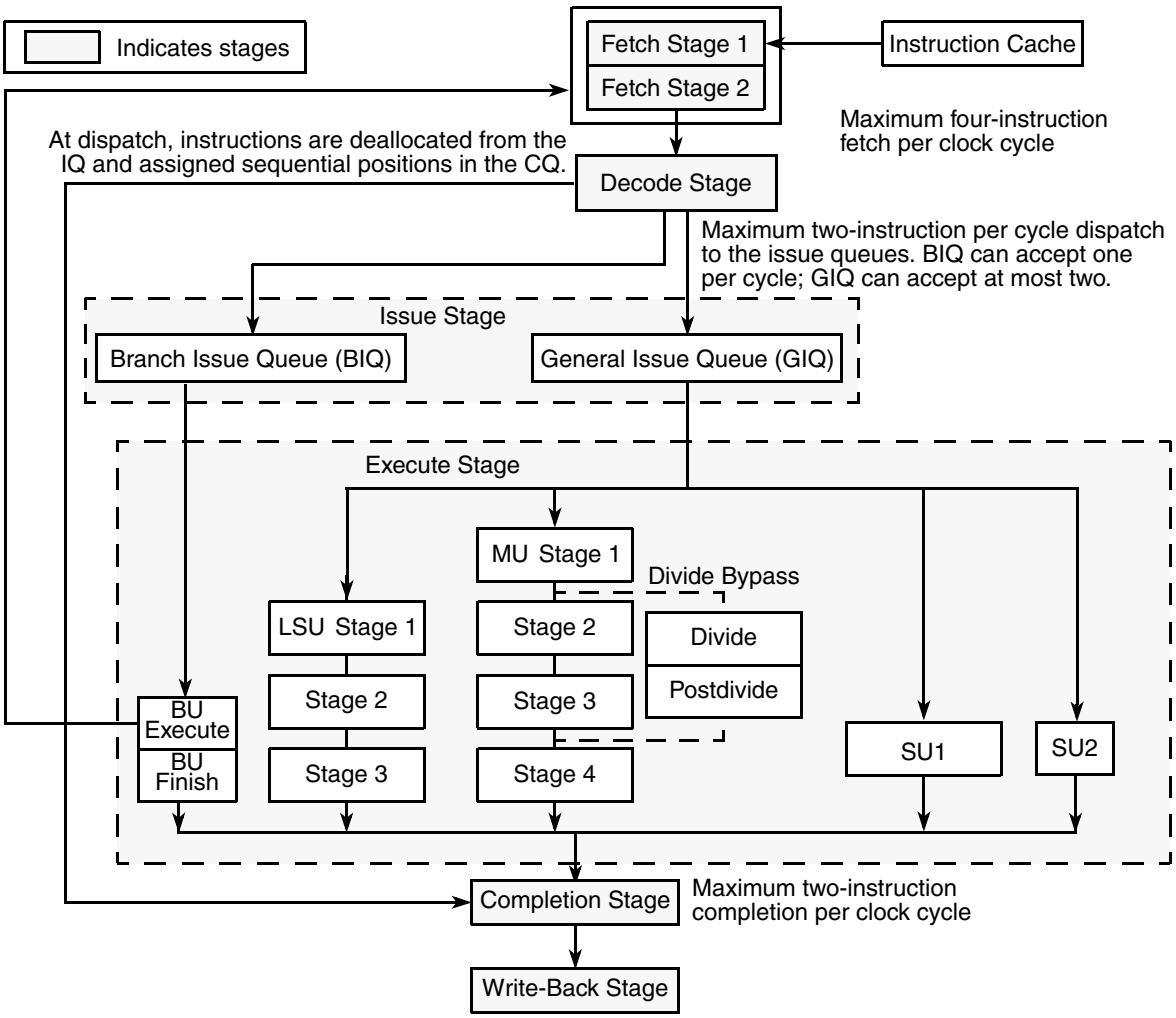


Figure 5-4. Instruction Pipeline Flow

The common pipeline stages are as follows:

- Instruction fetch—Includes the clock cycles necessary to request an instruction and the time the memory system takes to respond to the request. Instructions retrieved are latched into the instruction queue (IQ) for subsequent consideration by the dispatcher.

  Instruction fetch timing depends on many variables, such as whether an instruction is in the on-chip instruction cache or an L2 cache (if implemented). Those factors increase when it is necessary to fetch instructions from system memory and include the processor-to-bus clock ratio, the amount of bus traffic, and whether any cache coherency operations are required.

  Because there are so many variables, unless otherwise specified, the instruction timing examples in this chapter assume optimal performance and show the portion of the fetch stage in which the

instruction is in the instruction queue. The fetch1 and fetch2 stages are primarily involved in retrieving instructions.

- The decode/dispatch stage fully decodes each instruction; most instructions are dispatched to the issue queues (however, **isync**, **rfi**, **sc**, **nop**s, and some other instructions do not go to issue queues).
- The two issue queues, BIQ and GIQ, can accept as many as one and two instructions, respectively, in a cycle. The behavior of instruction dispatch is covered in significant detail in the *e500 Software Optimization Guide*. The following simplification covers most cases:
  - Instructions dispatch only from the two lowest IQ entries—IQ0 and IQ1.
  - A total of two instructions can be dispatched to the issue queues per clock cycle.
  - Space must be available in the CQ for an instruction to decode and dispatch (this includes instructions that are assigned a space in the CQ but not in an issue queue).

  Dispatch is treated as an event at the end of the decode stage. The issue stage reads source operands from rename registers and register files and determines when instructions are latched into the execution unit reservation stations. Note that the e500 has 14 rename registers, one for each completion queue entry, so instructions cannot stall because of a shortage of rename registers.

  The general behavior of the two issue queues is described as follows:
  - The GIQ accepts as many as two instructions from the dispatch unit per cycle. SU1, SU2, MU, and all LSU instructions (including 64-bit loads and stores) are dispatched to the GIQ, shown in Figure 5-5.

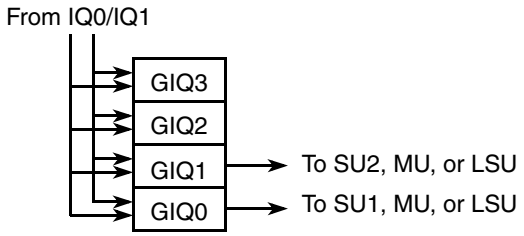


Figure 5-5. GPR Issue Queue (GIQ)

  Instructions can be issued out-of-order from the bottom two GIQ entries (GIQ1–GIQ0). GIQ0 can issue to SU1, MU, and LSU. GIQ1 can issue to SU2, MU, and LSU.

  Note that SU2 executes a subset of the instructions that can be executed in SU1. The ability to identify and dispatch instructions to SU2 increases the availability of SU1 to execute more computational-intensive instructions.

  An instruction in GIQ1 destined for SU2 or the LSU need not wait for an MU instruction in GIQ0 that is stalled behind a long-latency divide.
- The execute stage accepts instructions from its issue queue when the appropriate reservation stations are not busy. In this stage, the operands assigned to the execution stage from the issue stage are latched.

  The execution unit executes the instruction (perhaps over multiple cycles), writes results on its result bus, and notifies the CQ when the instruction finishes. The execution unit reports any exceptions to the completion stage. Instruction-generated exceptions are not taken until the excepting instruction is next to retire.

Most integer instructions have a 1-cycle latency, so results of these instructions are available 1 clock cycle after an instruction enters the execution unit. The MU and LSU are pipelined, as shown in Figure 5-4.

Branches resolve in execute stage. If a branch is mispredicted, it takes 5 cycles for the next instruction to reach the execute stage.

- The complete and write-back stages maintain the correct architectural machine state and commit results to the architecture-defined registers in the proper order. If completion logic detects an instruction containing an exception status or a mispredicted branch, all following instructions are cancelled, their execution results in rename registers are discarded, and the correct instruction stream is fetched.

  The complete stage ends when the instruction is retired. Two instructions can be retired per clock cycle. If no dependencies exist, as many as two instructions are retired in program order.

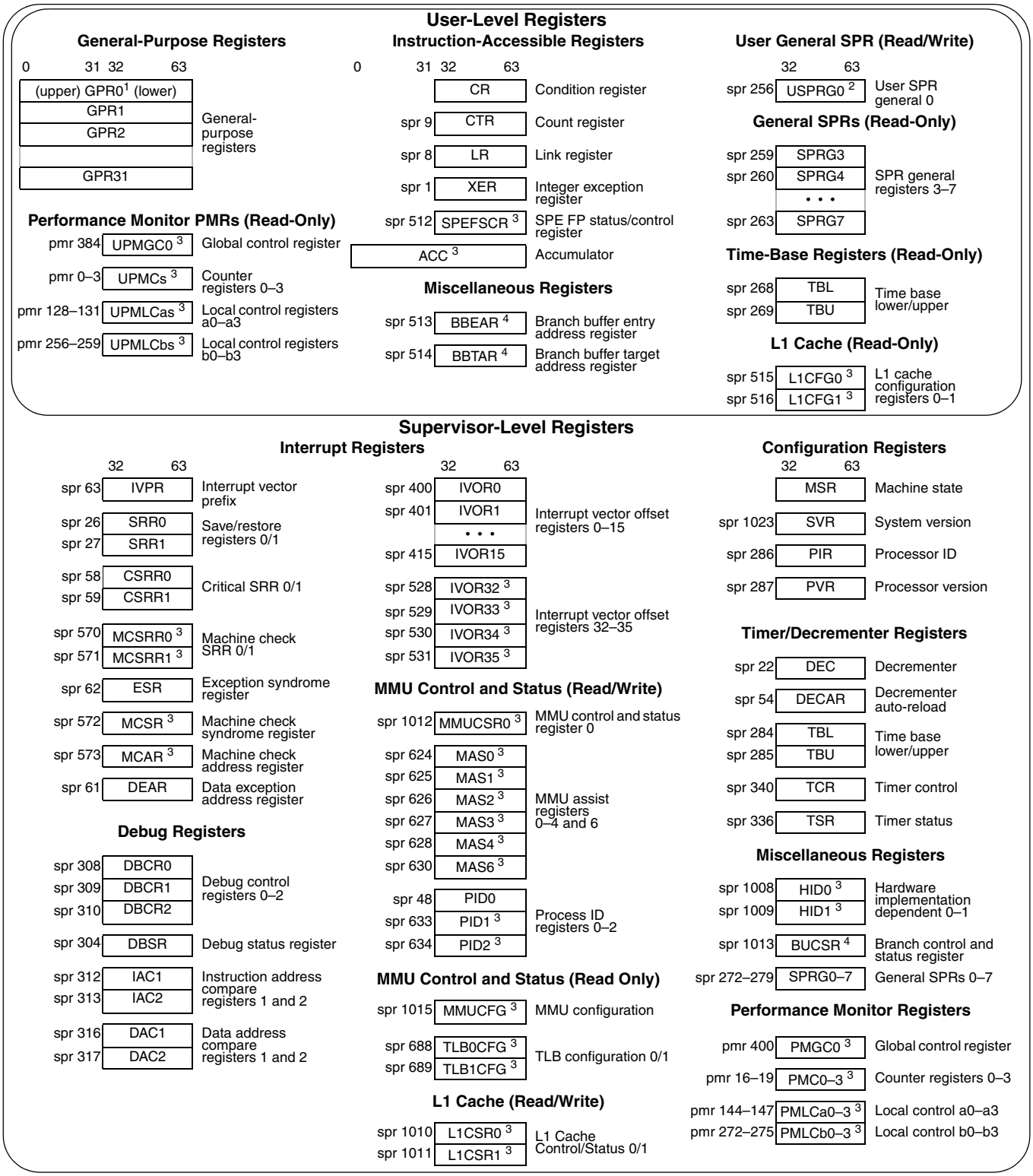
  The write-back stage occurs in the clock cycle after the instruction is retired.

The e500 core also provides new instructions that perform single-instruction, multiple-data (SIMD) operations. These signal processing instructions consist of parallel operations on both the upper and lower 32 bits of two 64-bit GPR values and produce two 32-bit results written to a 64-bit GPR.

As shown in Figure 5-4, the LSU, MU, and SU1 replicate logic to support 64-bit operations. Although a vector instruction generates separate, discrete results in the upper and lower halves of the target GPR, latency and throughput for vector instructions are the same as those for their scalar equivalents.

5.6 Programming Model

The following section describes the e500 core registers. Figure 5-6 shows the e500 register set.



1 The 64-bit GPR registers are accessed by the SPE as separate 32-bit registers by SPE instructions. Only SPE vector instructions can access the upper word.
2 USPRG0 is a separate physical register from SPRG0.
3 These registers are defined by the EIS.
4 These registers are e500-specific.

Figure 5-6. e500 Core Programming Model

5.7 On-Chip Cache Implementation

The core complex contains separate 32-Kbyte, eight-way set-associative, level 1 (L1) instruction and data caches to give rapid access to instructions and data.

The data cache supports four-state MESI memory coherency protocol. The core complex broadcasts all cache management functions based on the setting of the address broadcast enable bit, HID1[ABE], allowing management of other caches in the system.

On the MPC8555E the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

The caches implement a pseudo-least-recently-used (PLRU) replacement algorithm.

Parity generation and checking may be enabled for both caches, and each cache can be independently invalidated through L1CSR1 and L1CSR0. Additionally, instructions are provided to perform cache locking and unlocking on both data and instruction caches on a cache-block granularity. These are listed in Section 5.10.3, “Cache Control Instructions.”

Individual instruction cache blocks and data cache blocks can be invalidated using the **icbi** and **dcbi** instructions, respectively. The entire data cache can be invalidated by setting L1CSR0[CFI]; the entire instruction cache can be invalidated by setting L1CSR1[ICFI].

5.8 Interrupts and Exception Handling

The e500 core supports an extended exception handling model, with nested interrupt capability and extensive interrupt vector programmability. The following sections define the exception model, including an overview of exception handling as implemented on the e500 core, a brief description of the exception classes, and an overview of the registers involved in the processes.

5.8.1 Exception Handling

In general, interrupt processing begins with an exception that occurs due to external conditions, errors, or program execution problems. When the exception occurs, the processor checks to verify interrupt processing is enabled for that particular exception. If enabled, the interrupt causes the state of the processor to be saved in the appropriate registers and prepares to begin execution of the handler located at the associated vector address for that particular exception.

Once the handler is executing, the implementation may need to check one or more bits in the exception syndrome register (ESR) or the SPEFSCR, depending on the exception, to verify the specific cause of the exception and take appropriate action.

The core complex provides the interrupts described in Section 5.8.5, “Interrupt Registers.”

5.8.2 Interrupt Classes

All interrupts may be categorized as asynchronous/synchronous and critical/noncritical.

- Asynchronous interrupts (such as machine check, critical input, and external interrupts) are caused by events that are independent of instruction execution. For asynchronous interrupts, the address reported in a save/restore register is the address of the instruction that would have executed next had the asynchronous interrupt not occurred.
- Synchronous interrupts are those that are caused directly by the execution or attempted execution of instructions. Synchronous inputs may be either precise or imprecise, which are described as follows:
  - Synchronous precise interrupts are those that precisely indicate the address of the instruction causing the exception that generated the interrupt or, in some cases, the address of the immediately following instruction. The interrupt type and status bits indicate which instruction is addressed in the appropriate save/restore register.
  - Synchronous imprecise interrupts are those that may indicate the address of the instruction causing the exception that generated the interrupt or some instruction after the instruction causing the interrupt. If the interrupt was caused by either the context synchronizing mechanism or the execution synchronizing mechanism, the address in the appropriate save/restore register is the address of the interrupt forcing instruction. If the interrupt was not caused by either of those mechanisms, the address in the save/restore register is the last instruction to start execution and may not have completed. No instruction following the instruction in the save/restore register has executed.

5.8.3 Interrupt Types

The e500 core processes all interrupts as either machine check, critical, or noncritical types. Separate control and status register sets are provided for each interrupt type. The core handles interrupts from these three types in the following priority order:

1. Machine check interrupt (highest priority)—The e500 defines a separate set of resources for the machine check interrupt. They use the machine check save and restore registers (MCSRR0/MCSRR1) to save state when they are taken, and they use the **rfmci** instruction to restore state. These interrupts can be masked by the machine check enable bit, MSR[ME].
2. Noncritical interrupts—First-level interrupts that allow the processor to change program flow to handle conditions generated by external signals, errors, or unusual conditions arising from program execution or from programmable timer-related events. These interrupts are largely identical to those previously defined by the OEA portion of the architecture. They use save and restore registers (SRR0/SRR1) to save state when they are taken and they use the **rfi** instruction to restore state. Asynchronous noncritical interrupts can be masked by the external interrupt enable bit, MSR[EE].
3. Critical interrupts—Critical interrupts can be taken during a noncritical interrupt or during regular program flow. They use the critical save and restore registers (CSRR0/CSRR1) to save state when they are taken and they use the **rfci** instruction to restore state. These interrupts can be masked by

the critical enable bit, MSR[CE]. The embedded category defines the critical input, watchdog timer, and machine check interrupts as critical interrupts, but the e500 implements a third set of resources for the machine check interrupt, as described in Table 5-6.

All interrupts except machine check are ordered within the two categories of noncritical and critical, such that only one interrupt of each category is reported, and when it is processed (taken), no program state is lost. Because save/restore register pairs are serially reusable, program state may be lost when an unordered interrupt is taken.

5.8.4 Upper Bound on Interrupt Latencies

Core complex interrupt latency is defined as the number of core clocks between the sampling of the interrupt signal as asserted and the initiation of the IVOR fetch (that is, the fetch of the first instruction in the handler). Core complex interrupt latency is determinate unless a guarded load or a cache-inhibited **stwcx.** is being executed, in which case the latency is indeterminate. The minimum latency is 3 core clocks and the maximum is 8, not including the 2 bus clock cycles required to synchronize the interrupt signal from the pad.

When an interrupt is taken, all instructions in the IQ are thrown away unless the oldest instruction is a load/store instruction. That is, if an asynchronous interrupt is being serviced and the oldest instruction is not a load/store instruction, the core complex goes straight from sampling the interrupt to ensuring a recoverable state and issuing an exception. If a load/store instruction is oldest, the core complex waits 4 clocks before ensuring a recoverable state. During this time, any instruction finished by the LSU is deallocated.

5.8.5 Interrupt Registers

The registers associated with interrupt and exception handling are described in Table 5-6.

Table 5-6. Interrupt Registers

Register	Description
Noncritical Interrupt Registers	
SRR0	Save/restore register 0—Holds the address of the instruction causing the exception or the address of the instruction that will execute after the **rfi** instruction.
SRR1	Save/restore register 1—Holds machine state on noncritical interrupts and restores machine state after an **rfi** instruction is executed.
Critical Interrupt Registers	
CSRR0	Critical save/restore register 0—On critical interrupts, holds either the address of the instruction causing the exception or the address of the instruction that will execute after the **rfci** instruction.
CSRR1	Critical save/restore register 1—Holds machine state on critical interrupts and restores machine state after an **rfci** instruction is executed.
Machine Check Interrupt Registers	
MCSRR0	Machine check save/restore register 0—Used to store the address of the instruction that will execute after an **rfmci** instruction is executed.
MCSRR1	Machine check save/restore register 1—Holds machine state on machine check interrupts and restores machine state (if recoverable) after an **rfmci** instruction is executed.

Table 5-6. Interrupt Registers (continued)

Register	Description
MCAR	Machine check address register—Holds the address of the data or instruction that caused the machine check interrupt. MCAR contents are not meaningful if a signal triggered the machine check interrupt.
Syndrome Registers	
MCSR	Machine check syndrome register—Holds machine state information on machine check interrupts and restores machine state after an **rfmci** instruction is executed.
ESR	Exception syndrome register—Provides a syndrome to differentiate between the different kinds of exceptions that generate the same interrupt type. Upon generation of a specific exception type, the associated bit is set and all other bits are cleared.
SPE Interrupt Registers	
SPEFSCR	Signal processing and embedded floating-point status and control register—Provides interrupt control and status as well as various condition bits associated with the operations performed by the SPE.
Other Interrupt Registers	
DEAR	Data exception address register—Holds the address that was referenced by a load, store, or cache management instruction that caused an alignment, data TLB miss, or data storage interrupt.
IVPR IVORs	Together, IVPR[32–47] || IVOR*n* [48–59] || 0b0000 define the address of an interrupt-processing routine. See Table 5-7 and the EREF for more information.

Each interrupt has an associated interrupt vector address, obtained by concatenating the IVPR value with the address index in the associated IVOR (that is, IVPR[32–47] || IVOR*n*[48–59] || 0b0000). The resulting address is that of the instruction to be executed when that interrupt occurs. IVPR and IVOR values are indeterminate on reset, and must be initialized by the system software using **mtspr**. Table 5-7 lists IVOR registers implemented on the e500 and the associated interrupts.

Table 5-7. Interrupt Vector Registers and Exception Conditions

Register	Interrupt
Embedded Category–Defined IVORs	
IVOR0	Critical input
IVOR1	Machine check interrupt offset
IVOR2	Data storage interrupt offset
IVOR3	Instruction storage interrupt offset
IVOR4	External input interrupt offset
IVOR5	Alignment interrupt offset
IVOR6	Program interrupt offset
IVOR7	Floating-point unavailable interrupt offset
IVOR8	System call interrupt offset
IVOR9	Auxiliary processor unavailable interrupt offset
IVOR10	Decrementer interrupt offset
IVOR11	Fixed-interval timer interrupt offset
IVOR12	Watchdog timer interrupt offset
IVOR13	Data TLB error interrupt offset
IVOR14	Instruction TLB error interrupt offset
IVOR15	Debug interrupt offset

Table 5-7. Interrupt Vector Registers and Exception Conditions (continued)

Register	Interrupt
e500-Specific IVORs	
IVOR32	SPE unavailable interrupt offset
IVOR33	SPE floating-point data exception interrupt offset
IVOR34	SPE floating-point round exception interrupt offset
IVOR35	Performance monitor

5.9 Memory Management

The e500 core complex supports demand-paged virtual memory as well other memory management schemes that depend on precise control of effective-to-physical address translation and flexible memory protection as defined by the architecture. The mapping mechanism consists of software-managed TLBs that support variable-sized pages with per-page properties and permissions. The following properties can be configured for each TLB:

- User-mode page execute access
- User-mode page read access
- User-mode page write access
- Supervisor-mode page execute access
- Supervisor-mode page read access
- Supervisor-mode page write access
- Write-through required (W)
- Caching inhibited (I)
- Memory coherency required (M) (ignored on the MPC8555E)
- Guarded (G)
- Endianness (E)
- User-definable (U0–U3), a 4-bit implementation-specific field

The core complex employs a two-level memory management unit (MMU) architecture. There are separate instruction and data level-1 (L1) MMUs backed up by a unified level-2 (L2) MMU.

This two-level structure is shown in Figure 5-7.

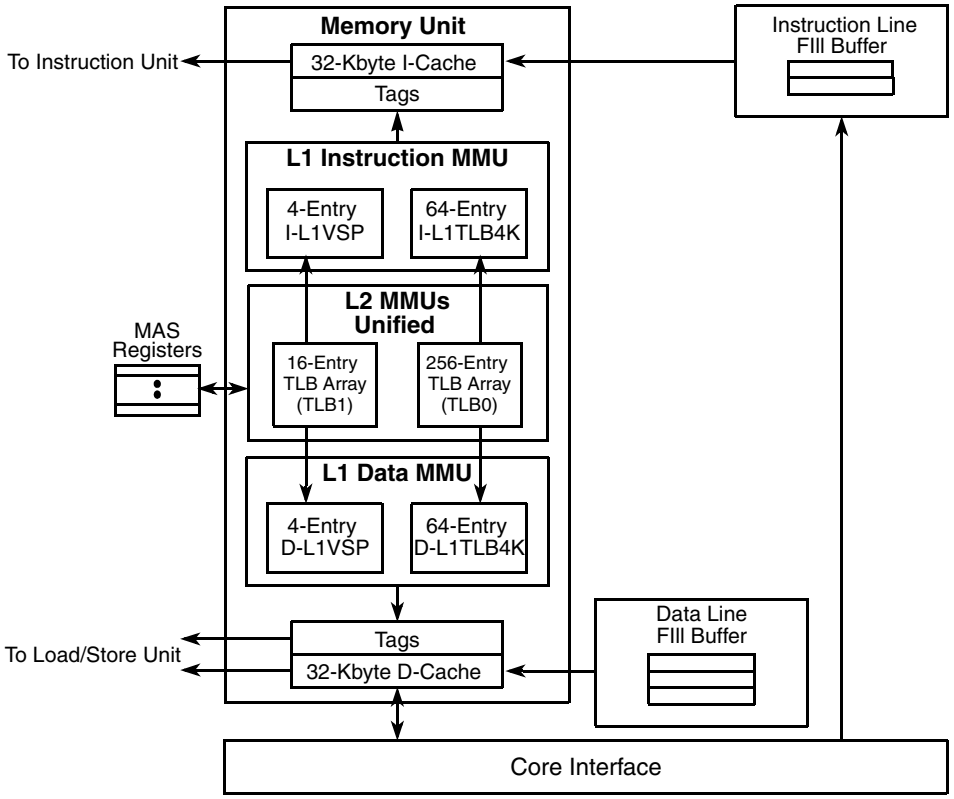


Figure 5-7. MMU Structure

Level-1 MMUs have the following features:

- Four-entry, fully associative TLB array that supports all nine page sizes
- 64-entry, 4-way set-associative TLB 4-Kbyte array that supports 4-Kbyte pages only
- Hardware partially managed by L2 MMU
- Supports snooping of TLBs by both internal and external **tlbivax** instructions

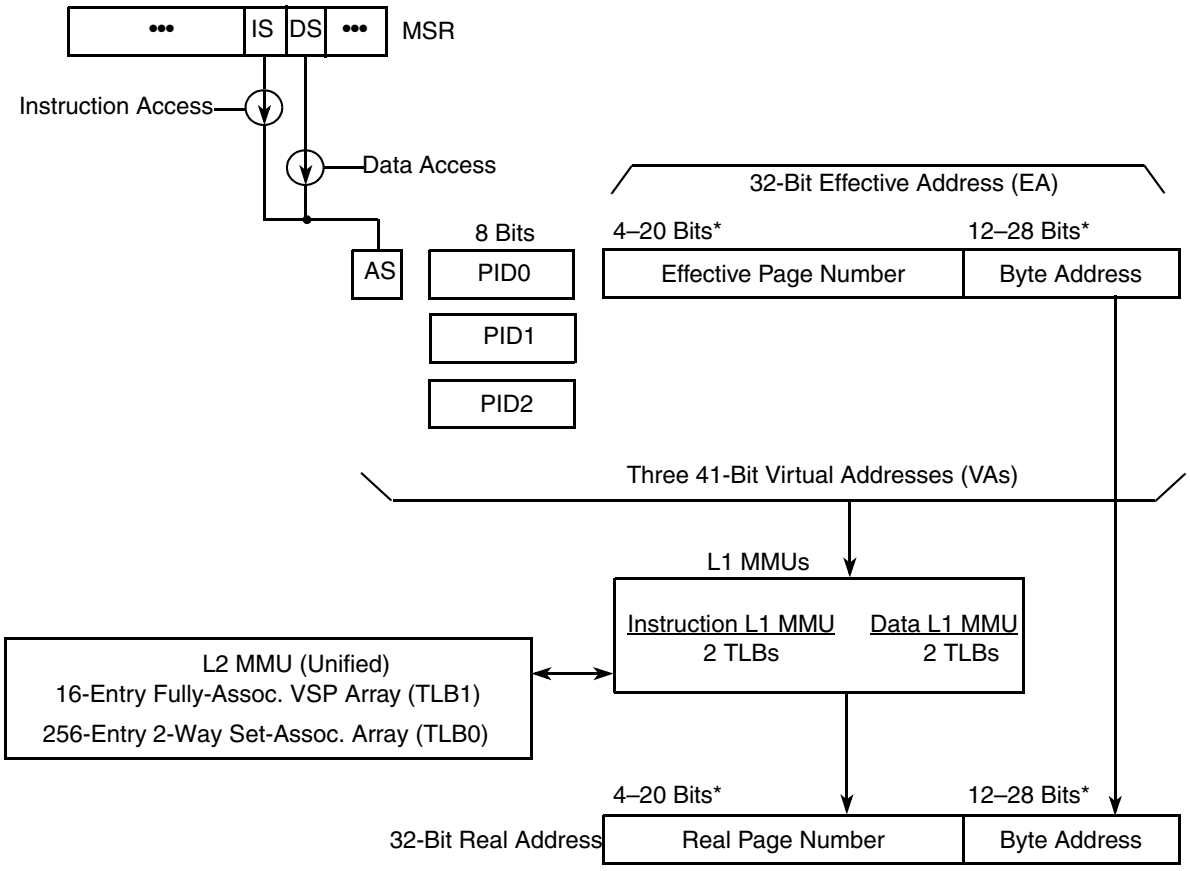
The level-2 MMU has the following features:

- A 16-entry, fully associative L2 TLB array (TLB1) that supports all nine variable page sizes
- TLB array (TLB0) that supports only 4-Kbyte pages, as follows:
  — 256-entry, 2-way set-associative TLB array
- Hardware assist for TLB miss exceptions
- Software managed by **tlbre**, **tlbwe**, **tlbsx**, **tlbsync**, **tlbivax**, and **mtspr** instructions
- Supports snooping of TLB by both internal and external **tlbivax** instructions

5.9.1 Address Translation

The core complex fetch and load/store units generate 32-bit effective addresses. The MMU translates these addresses to real addresses (which are used for memory bus accesses) using an interim 41-bit virtual address.

Figure 5-8 shows the translation flow.



* Number of bits depends on page size (4 Kbytes–256 Mbytes).

Figure 5-8. Effective-to-Real Address Translation Flow

The appropriate L1 MMU (instruction or data) is checked for a matching address translation. The instruction L1 MMU and data L1 MMU operate independently and can be accessed in parallel, so that hits for instruction accesses and data accesses can occur in the same clock. If an L1 MMU misses, the request for translation is forwarded to the unified (instruction and data) L2 MMU. If found, the contents of the TLB entry are concatenated with the byte address to obtain the physical address of the requested access. On misses, the L1 TLB entries are replaced from their L2 TLB counterparts using a true LRU algorithm.

5.9.2 MMU Assist Registers (MAS0–MAS4 and MAS6)

MMU assist registers are used to hold values either read from or to be written to the TLBs and information required to identify the TLB to be accessed. MAS3 implements the real page number (RPN), the user attribute bits (U0–U3), and permission bits (UX, SX, UW, SW, UR, SR) that specify user and supervisor read, write, and execute permissions.

The e500 does not implement MAS5.

MAS registers are affected by the following instructions (see the EREF for more detailed information):

- MAS registers are accessed with the **mtspr** and **mfspr** instructions.

- The TLB Read Entry instruction (**tlbre**) causes the contents of a single TLB entry from the L2 MMU to be placed in defined locations in MAS0–MAS3 (and optionally MAS7 on the e500v2). The TLB entry to be extracted is determined by information written to MAS0 and MAS2 before the **tlbre** instruction is executed.
- The TLB Write Entry instruction (**tlbwe**) causes the information stored in certain locations of MAS0–MAS3 (and MAS7 on the e500v2) to be written to the TLB specified in MAS0.
- The TLB Search Indexed instruction (**tlbsx**) updates MAS registers conditionally, based on success or failure of a lookup in the L2 MMU. The lookup is specified by the instruction encoding and specific search fields in MAS6. The values placed in the MAS registers may differ, depending on a successful or unsuccessful search.

For TLB miss and certain MMU-related DSI/ISI exceptions, MAS4 provides default values for updating MAS0–MAS2.

5.9.3 Process ID Registers (PID0–PID2)

The e500 core complex also implements three process ID (PID) registers that hold the values used to construct the three virtual addresses for each access. These process IDs provide an extended page sharing capability. Which of these three virtual addresses is used is controlled by the TID field of a matching TLB entry, and when TID = 0x00 (identifying a page as globally shared), the PID values are ignored.

A hit to multiple TLB entries in the L1 MMU (even if they are in separate arrays) or a hit to multiple entries in the L2 MMU is considered to be a programming error.

5.9.4 TLB Coherency

The core complex provides the ability to invalidate a TLB entry, as defined by the architecture. The **tlbivax** instruction invalidates a matching local TLB entry. Execution of this instruction is also broadcast on the core complex bus (CCB) if HID1[ABE] is set. The core complex also snoops TLB invalidate transactions on the CCB from other bus masters.

On the MPC8555E the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

5.10 Memory Coherency

The core complex supports four-state memory coherency. Memory coherency is hardware-supported on the system bus through bus snooping and the retry/copyback bus protocol, and through broadcasting of cache management instructions. Translation coherency is also hardware-supported through broadcasting and bus snooping of TLB invalidate transactions. The four-state MESI protocol supports efficient large-scale real-time data sharing between multiple caching bus masters.

5.10.1 Atomic Update Memory References

The e500 core supports atomic update memory references for both aligned word forms of data using the load and reserve and store conditional instruction pair, **lwarx** and **stwcx**. Typically, a load and reserve

instruction establishes a reservation and is paired with a store conditional instruction to achieve the atomic operation. However, there are restrictions and requirements for this functionality. The processor revokes reservations during a context switch, so the programmer must reacquire the reservation after a context switch occurs.

5.10.2 Memory Access Ordering

The core complex supports weakly ordered references to memory. Thus the e500 manages the order and synchronization of instructions to ensure proper execution when memory is shared between multiple processes or programs. The cache and data memory control attributes, along with **msync** and **mbar**, provide the required access control.

5.10.3 Cache Control Instructions

The core complex supports instructions for performing a full range of cache control functions, including cache locking by line. The core complex supports broadcasting and snooping of these cache control instructions on the CCB. The e500 core also supports the following e500-specific cache locking instructions:

- Data Cache Block Lock Clear (**dcblc**)
- Data Cache Block Touch and Lock Set (**dcbtls**)
- Data Cache Block Touch for Store and Lock Set (**dcbtstls**)
- Instruction Cache Block Lock Clear (**icblc**)
- Instruction Cache Block Touch and Lock Set (**icbtls**)

5.10.4 Programmable Page Characteristics

Cache and memory attributes are programmable on a per-page basis. In addition to the write-through, caching-inhibited, memory coherency enforced, and guarded characteristics defined by the WIMG bits, the endianness bit, E, allows selection of big- or little-endian byte ordering on a per-page basis.

In addition to the WIMGE bits, the MMU model defines user-definable page attribute bits U0–U3.

5.11 Core Complex Bus (CCB)

The core complex defines a versatile local bus interface that allows a wide range of system performance and system-complexity trade-offs. The interface defines the following buses:

- An address-out bus for mastering bus transactions
- An address-in bus for snooping internal resources
- Three tagged data buses

Two of the data buses are general-purpose data-in buses for reads, and the third is a data-out bus for writes. The two data-in buses feature support for out-of-order read transactions from two different sources simultaneously, and all three data buses may be operated concurrently. The address-in bus supports snooping for external management of the L1 caches and TLBs by other bus masters. The core complex

broadcasts and snoops the cache and TLB management instructions accordingly. It is envisioned that a wide range of system implementations can be constructed from the defined interface.

5.12 Performance Monitoring

The e500 core provides a performance monitoring capability that allows counting of events such as processor clocks, instruction cache misses, data cache misses, mispredicted branches, and others. The count of these events may be configured to trigger a performance monitor exception following the e500 interrupt model. This interrupt is assigned to vector offset register IVOR35.

The register set associated with the performance monitoring function consists of counter registers, a global control register, and local control registers. These registers are read/write from supervisor mode, and each register is reflected to a corresponding read-only register for user mode. Two instructions, **mtpmr** and **mfpmr**, are provided for moving data to and from these registers. An overview of the performance monitoring registers is provided in the following sections.

5.12.1 Global Control Register

The PMGC0 register provides global control of the performance monitoring facility from supervisor mode. From this register all counters may be frozen, unfrozen, or configured to freeze on an enabled condition or event. Additionally, the performance monitoring facility may be disabled or enabled from this register. The contents of PMGC0 are reflected to UPMGC0, which may be read from user mode using the **mfpmr** instruction.

5.12.2 Performance Monitor Counter Registers

There are four counter registers (PCM0–PCM3) provided in the performance monitoring facility. These 32-bit registers hold the current count for software-selectable events and can be programmed to generate an exception on overflow. These registers may be written or read from supervisor mode using the **mtpmr** and **mfpmr** instructions. The contents of these registers are reflected to UPCM0–UPCM3, which can be read from user mode with **mfpmr**.

Performance monitor exceptions occur only if all of the following conditions are met:

- A counter is in the overflow state.
- The counter's overflow signaling is enabled.
- Overflow exception generation is enabled in PMGC0.
- MSR[EE] is set.

5.12.3 Local Control Registers

For each of the counter registers, there are two corresponding local control registers. These two registers specify which of the 128 available events is to be counted, what specific action is to be taken on overflow, and various options for freezing a counter value under given modes or conditions.

- PMLCa0–PMLCa3 provide fields that allow freezing of the corresponding counter in user mode, supervisor mode, or under software control. Additionally, the overflow condition may be enabled

or disabled from this register. The contents of these registers are reflected to UPMLCa0–UPMLCa3, which can be read from user mode with **mfpmr**.

- PMLCb0–PMLCb3 provide count scaling for each counter register using configurable threshold and multiplier values. The threshold is a 6-bit value and the multiplier is a 3-bit encoded value, allowing eight multiplier values in the range of 1 to 128. Any counter may be configured to increment only when an event occurs more than [threshold × multiplier] times. The contents of these registers are reflected to UPMLCb0–UPMLCb3, which can be read from user mode with **mfpmr**.

5.13 Legacy Support of Power Architecture Technology

This section provides an overview of the architectural differences and compatibilities of the e500 core compared with the AIM Power Architecture technology. The two levels of the e500 programming environment are as follows:

- User level—This defines the base user-level instruction set, user-level registers, data types, memory conventions, and the memory and programming models seen by application programmers.
- Supervisor level—This defines supervisor-level resources typically required by an operating system, the memory management model, supervisor level registers, and the exception model.

Like all devices that implement the Power Architecture technology, in general, the e500 core supports the user-level architecture. The following sections are intended to highlight the main differences. For specific implementation details refer to the relevant chapter.

5.13.1 Instruction Set Compatibility

The following sections generally describe the user and supervisor instruction sets.

5.13.1.1 User Instruction Set

The e500 core executes legacy user-mode binaries and object files except for the following:

- The e500 supports vector and scalar single-precision floating-point operations as part of the SPE. The e500v2 supports scalar double-precision floating-point instructions. These instructions have different encoding than the AIM definition of the architecture. Additionally, the e500 core uses GPRs for floating-point operations, rather than the FPRs defined by the UISA. Most porting of floating-point operations can be handled by recompiling.
- String instructions are not implemented on the e500; therefore, trap emulation must be provided to ensure backward compatibility.

5.13.1.2 Supervisor Instruction Set

The supervisor mode instruction set defined by the PowerPC architecture is compatible with the e500 with the following exceptions:

- The MMU architecture is different, so some TLB manipulation instructions have different semantics.

- Instructions that support the BATs and segment registers are not implemented.

5.13.2 Memory Subsystem

The architecture provides separate instruction and data memory resources. The e500 provides additional cache control features, including cache locking.

5.13.3 Exception Handling

Exception handling is generally the same as that defined in the AIM version of the architecture for the e500, with the following differences:

- The critical interrupt provides an extra level of interrupt nesting. The critical interrupt includes external critical and watchdog timer time-out inputs.
- The machine check exception uses the Return from Machine Check Interrupt instruction, **rfmci**, and two machine check save/restore registers, MCSRR0 and MCSRR1.
- IVPR and IVORs set interrupt vectors individually, but they can be set to the address offsets defined in the OEA to provide compatibility.
- The embedded category does not define a reset vector; execution begins at a fixed virtual address, 0xFFFF_FFFC.
- Timer services are generally compatible, although the embedded category defines a new decrementer auto reload feature, the fixed-interval timer critical interrupt, and the watchdog timer interrupt, which are implemented in the e500 core.

An overview of the interrupt and exception handling capabilities of the e500 core can be found in Section 5.8, “Interrupts and Exception Handling.”

5.13.4 Memory Management

The embedded category defines resources for fixed 4-Kbyte pages and multiple, variable page sizes that can be configured in a single implementation. TLB management is provided with new instructions and SPRs.

5.13.5 Reset

Embedded category–compliant cores do not share a common reset vector with the AIM version of the architecture. Instead, at reset fetching begins at address 0xFFFF_FFFC. In addition, the Freescale MMU category defines specific aspects of the MMU page translation and protection mechanisms. Unlike the AIM version of the core, as soon as instruction fetching begins, the e500 core is in virtual mode with a hardware-initialized TLB entry.

MMU operations are described in the EREF.

5.13.6 Little-Endian Mode

Unlike the AIM version of the architecture, where little-endian mode is controlled on a system basis, the embedded category allows control of byte ordering on a memory page basis. In addition, the little-endian byte ordering used is true little endian.

5.14 PowerQUICC III Implementation Details

Table 5-8 summarizes e500 core functionality that is not implemented by PowerQUICC III devices.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation

Feature	PowerQUICC III Implementation
Cache protocol	The L2 cache does not support MESI cache protocol.
Multiprocessor functionality	Because PowerQUICC III is designed for a uniprocessor environment, the following e500 functionality is not implemented: • The memory coherence bit, M, which is controlled through MAS2[M] and MAS4[MD] has no effect. • HID1[ABE] has meaning only in that it must be set to ensure that cache and TLB management instructions operate properly with respect to the L2 cache. • Dynamic snooping does not occur in power-stopped state (see the note below in the entry for dynamic bus snooping).
Nexus support	Nexus is not supported. The Nexus processor ID register (NPIDR) and the Nexus bus enable bit (HID1[NEXEN]) are not supported.
R1 and R2 data bus parity	R1 and R2 data bus parity are disabled on PowerQUICC III devices. HID1[R1DPE,R2DPE] are reserved.
Dynamic bus snooping	The PowerQUICC III devices do not perform dynamic bus snooping as described here. That is, when the e500 core is in core-stopped state (which is the state of the core when the PowerQUICC III device is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, L1 caches should be flushed if coherency is required during these power-down modes. For more information, see Section 18.5.1.9, “Snooping in Power-Down Modes.”
Supported TCR[WRC]	PowerQUICC III devices define values for 01, 10, and 11, as follows: 00 No watchdog timer reset can occur. 01 Force processor checkstop on second timeout of watchdog timer 10 Assert processor reset output (*core_hreset_req*) on second timeout of watchdog timer 11 Reserved
SPE and floating-point categories	The SPE and the vector and scalar floating-point instructions will not be implemented in the next generation of PowerQUICC devices. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses these instructions at the assembly level or that uses SPE or floating-point intrinsics will require rewriting for upward compatibility with next generation PowerQUICC devices. Freescale Semiconductor offers a libcfsl_e500 library that uses SPE instructions. Freescale Semiconductor will also provide future libraries to support next generation PowerQUICC devices.
HID0 implementation	SEL_TBCLK bit. Selects time base clock. If this bit is set and the time base is enabled, the time base is based on the TBCLK input, which on the PowerQUICC III devices is RTC.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation (continued)

Feature	PowerQUICC III Implementation
HID1 Implementation	PLL_MODE. Set to 01 PLL_CFG. PowerQUICC III devices support the following: 0001_00 Ratio of 2:1 0001_01 Ratio of 5:2 (2.5:1) 0001_10 Ratio of 3:1 0001_11 Ratio of 7:2 (3.5:1) NEXEN, R1DPE, R2DPE, MPXTT, MSHARS, SSHAR, ATS, and MID are not implemented On PowerQUICC III devices, ABE must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.
	Please refer to the description of HID1[RFXE] in Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." If RFXE is 0, conditions that cause the assertion of $\overline{\textit{core\_fault\_in}}$ cannot directly cause the e500 to generate a machine check; however, PowerQUICC III devices must be configured to detect and enable such conditions. The following describes how error bits should be configured: • ECM mapping errors: EEER[LAEE] must be set. See Section 8.2.1.4, "ECM Error Enable Register (EEER)." • L2 multiple-bit ECC errors: L2ERRDIS[MBECCDIS] must be cleared to ensure that error can be detected. L2ERRINTEN[MBECCINTEN] must be set. See Section 7.3.1.5, "L2 Error Registers." • DDR multiple-bit ECC errors. ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and DDR_SDRAM_CFG[ECC_EN] must be one to ensure that an interrupt is generated. See Section 9.4.1, "Register Descriptions." • PCI. The appropriate parity detect and master-abort bits in ERR_DR must be cleared and the corresponding enable bits in ERR_EN must be set to ensure that an interrupt is generated. Local bus controller parity errors. LTEDR[PARD] must be cleared and LTEIR[PARI] must be set to ensure that an parity errors can generate an interrupt. See Section 13.3.1.11, "Transfer Error Check Disable Register (LTEDR)," and Section 13.3.1.12, "Transfer Error Interrupt Enable Register (LTEIR)."
PIR value	The PIR value is all zeros on PowerQUICC III devices.
PVR value	The PVR reset value is 0x80*nn_nnnn.* See Table 5-1 for specific values. PVR[VERSION] = 0x80*nn* PVR[REVISION] = 0x*nnnn*
SVR value	The SVR reset value is 0x80*nn_nnnn.* See Table 5-1 for specific values.
Alternate time base	The alternate time base defines a time base counter similar to the time base defined in architecture. It is intended to be used for measuring time in implementation defined intervals. It differs from the defined Time Base in that it is not writable and always counts up, wrapping when the 64-bit count overflows. It defines two SPRs, ATBL (SPR 526) and ATBL (SPR 527).

Chapter 6
Core Register Summary

This chapter describes the e500 register model and indicates whether each register is defined by the Power Architecture technology, by the Freescale embedded category implementation standards (EIS), or by the implementation. For the programmer, drawing this distinction indicates the degree to which code is portable among Freescale processors.

This chapter provides reference material—figures for each register and complete descriptions of register fields, including how the registers are accessed, reset values, and whether they can be accessed by user- and supervisor-level software. Detailed discussions of how these registers are used are provided in *EREF: A Reference for Freescale Book E and the e500 Core* and the *PowerPC™ e500 Core Family Reference Manual*.

Note that all registers described here are implemented in the hardware as part of the e500 core.

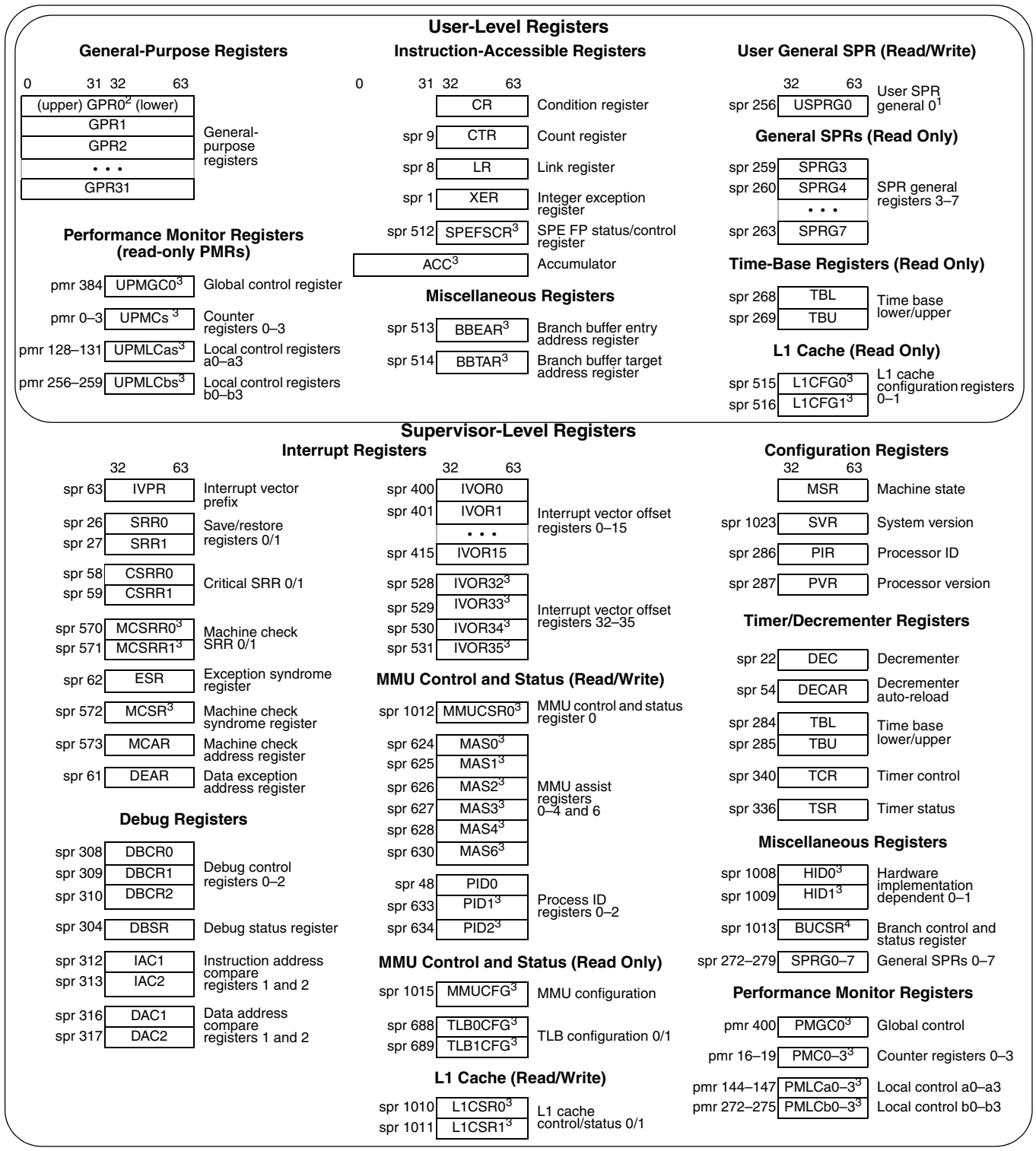
6.1 Overview

As shown in Figure 6-1, most of the registers implemented are defined by the architecture, and most of those were defined by the AIM definition of the architecture and have changed very little. Additional registers and fields within registers are defined by the EIS and by the implementation.

The Power Architecture technology defines some register fields in a very general way, leaving some details as implementation specific. In some cases, this more specific functionality is defined by the EIS; in others it is left up to the processor. This chapter identifies the level at which each features is defined.

6.1.1 Register Set

Table 6-1 shows the e500 register set, grouped by whether they can be accessed by user- or supervisor-level software. Unless otherwise indicated, these registers are defined by the base or embedded category.



1 USPRG0 is a separate physical register from SPRG0.
2 The 64-bit GPR registers are accessed by the SPE as separate 32-bit registers by SPE instructions. Only SPE vector instructions can access the upper word.
3 These registers are defined by additional categories.
4 These registers are e500-specific.

Figure 6-1. Core Register Model

6.2 Register Model for 32-Bit Implementations

Embedded 32-bit processors implement the following types of software-accessible registers:

- Architecture-defined registers that are accessed as part of instruction execution. These include the following:
  - — Registers used for computation. These include the following:
    - – General-purpose registers (GPRs)—The 32 GPRs hold source and destination operands for load, store, arithmetic, and computational instructions, and to read and write to other registers. The e500 implements these as 64-bit registers for use with 64-bit load, store, and merge instructions, as described in Section 6.3.1, “General-Purpose Registers (GPRs).”
    - – Integer exception register (XER)—Bits in this register are set based on the operation of an instruction considered as a whole, not on intermediate results. (For example, the Subtract from Carrying instruction (**subfc**), the result of which is specified as the sum of three values, sets bits in the XER based on the entire operation, not on an intermediate sum.)

    These registers are described in Section 6.3, “Registers for Computational Operations.”
  - — Condition register (CR)—Used to record conditions such as overflows and carries that occur as a result of executing arithmetic instructions (including those implemented by the SPE). The CR is described in Section 6.4, “Registers for Branch Operations.”
  - — Machine state register (MSR)—Used by the operating system to configure parameters such as user/supervisor mode, address space, and enabling of asynchronous interrupts. This register is described in Section 6.5.1, “Machine State Register (MSR),” grouped with processor control SPRs.
- Special-purpose registers (SPRs) are accessed explicitly using **mtspr** and **mfspr** instructions. These registers are listed in Table 6-1 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Freescale EIS– and e500-defined SPRs that are accessed explicitly using **mtspr** and **mfspr** are listed in Table 6-2 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Freescale EIS–defined performance monitor registers (PMRs). These registers are similar to SPRs, but are accessed with Freescale EIS–defined move to and move from PMR instructions (**mtpmr** and **mfpmr**).

In this chapter, SPRs are grouped by function as follows:

- Section 6.4, “Registers for Branch Operations,” describes the count register (CTR) and the link register (LR).
- Section 6.5, “Processor Control Registers”
- Section 6.6, “Timer Registers”
- Section 6.7, “Interrupt Registers”
- Section 6.8, “Software-Use SPRs (SPRG0–SPRG7 and USPRG0),” describes SPRs defined for software use.
- Section 6.9, “Branch Target Buffer (BTB) Registers,” describes e500-specific registers defined to support the e500 tabs.
- Section 6.10, “Hardware Implementation-Dependent Registers,” describes HID0 and HID1.
- Section 6.11, “L1 Cache Configuration Registers”

- Section 6.12, “MMU Registers”
- Section 6.13, “Debug Registers”
- Section 6.14, “Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)”

The e500 core implements 64-bit GPRs, the upper 32 bits of which are used only with 64-bit load, store, and merge instructions.

6.2.1 Special-Purpose Registers (SPRs)

Table 6-1 summarizes SPRs. The SPR numbers are used in the instruction mnemonics. Bit 5 in an SPR number indicates whether an SPR is accessible from user- or supervisor-level software. An **mtspr** or **mfspr** instruction that specifies an unsupported SPR number is considered an invalid instruction.

In Table 6-1 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.

NOTE

Writing to the following registers requires synchronization, as described in the “Synchronization Requirements” section in the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*.

- BTB locking registers—BBEAR, BBTAR, and BUCSR
- DBCR*n*
- HID*n*
- L1CSR*n*
- MMU registers—MAS*n*, MMUCSR0, PID*n*
- SPEFSCR

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
CSRR0	Critical save/restore register 0	58	00001 11010	Read/Write	Yes	6.7.1.1/6-17
CSRR1	Critical save/restore register 1	59	00001 11011	Read/Write	Yes	6.7.1.2/6-17
CTR	Count register	9	00000 01001	Read/Write	No	6.4.3/6-11

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
DAC1	Data address compare 1	316	01001 11100	Read/Write	Yes	6.13.4/6-44
DAC2	Data address compare 2	317	01001 11101			
DBCR0	Debug control register 0 [1]	308	01001 10100	Read/Write	Yes	6.13.1/6-39
DBCR1	Debug control register 1 [1]	309	01001 10101	Read/Write	Yes	
DBCR2	Debug control register 2 [1]	310	01001 10110	Read/Write	Yes	
DBSR	Debug status register	304	01001 10000	w1c[2]	Yes	6.13.2/6-42
DEAR	Data exception address register	61	00001 11101	Read/Write	Yes	6.7.1.5/6-18
DEC	Decrementer	22	00000 10110	Read/Write	Yes	6.6.4/6-16
DECAR	Decrementer auto-reload	54	00001 10110	Write only		
ESR	Exception syndrome register	62	00001 11110	Read/Write	Yes	6.7.1.8/6-19
IAC1	Instruction address compare 1	312	01001 11000	Read/Write	Yes	6.13.3/6-44
IAC2	Instruction address compare 2	313	01001 11001			
IVOR0	Critical input	400	01100 10000	Read/Write	Yes	6.7.1.7/6-18
IVOR1	Machine check interrupt offset	401	01100 10001			
IVOR2	Data storage interrupt offset	402	01100 10010			
IVOR3	Instruction storage interrupt offset	403	01100 10011			
IVOR4	External input interrupt offset	404	01100 10100			
IVOR5	Alignment interrupt offset	405	01100 10101			
IVOR6	Program interrupt offset	406	01100 10110			
IVOR8	System call interrupt offset	408	01100 11000			
IVOR10	Decrementer interrupt offset	410	01100 11010			
IVOR11	Fixed-interval timer interrupt offset	411	01100 11011			
IVOR12	Watchdog timer interrupt offset	412	01100 11100			
IVOR13	Data TLB error interrupt offset	413	01100 11101			
IVOR14	Instruction TLB error interrupt offset	414	01100 11110			
IVOR15	Debug interrupt offset	415	01100 11111			
IVPR	Interrupt vector	63	00001 11111	Read/Write	Yes	6.7.1.6/6-18
LR	Link register	8	00000 01000	Read/Write	No	6.4.2/6-11
PID	Process ID register [3]	48	00001 10000	Read/Write	Yes	6.12.1/6-32
PIR	Processor ID register	286	01000 11110	Read only	Yes	6.5.2/6-13
PVR	Processor version register	287	01000 11111	Read only	Yes	6.5.3/6-13

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
SPRG0	SPR general 0	272	01000 10000	Read/Write	Yes	6.8/6-22
SPRG1	SPR general 1	273	01000 10001	Read/Write	Yes	
SPRG2	SPR general 2	274	01000 10010	Read/Write	Yes	
SPRG3	SPR general 3	259	01000 00011	Read only	No[4]	
		275	01000 10011	Read/Write	Yes	
SPRG4	SPR general 4	260	01000 00100	Read only	No	
		276	01000 10100	Read/Write	Yes	
SPRG5	SPR general 5	261	01000 00101	Read only	No	6.8/6-22
		277	01000 10101	Read/Write	Yes	
SPRG6	SPR general 6	262	01000 00110	Read only	No	
		278	01000 10110	Read/Write	Yes	
SPRG7	SPR general 7	263	01000 00111	Read only	No	
		279	01000 10111	Read/Write	Yes	
SRR0	Save/restore register 0	26	00000 11010	Read/Write	Yes	6.7.1.1/6-17
SRR1	Save/restore register 1	27	00000 11011	Read/Write	Yes	6.7.1.2/6-17
TBL	Time base lower	268	01000 01100	Read only	No	6.6.3/6-16
		284	01000 11100	Write only	Yes	
TBU	Time base upper	269	01000 01101	Read only	No	
		285	01000 11101	Write only	Yes	
TCR	Timer control register	340	01010 10100	Read/Write	Yes	6.6.1/6-14
TSR	Timer status register	336	01010 10000	w1c[5]	Yes	6.6.2/6-15
USPRG0	User SPR general 0[6]	256	01000 00000	Read/Write	No	6.8/6-22
XER	Integer exception register	1	00000 00001	Read/Write	No	6.3.2/6-8

1 Accesses to this register requires synchronization, as described in the “Synchronization Requirements” section of the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*

2 The DBSR is read using **mfspr**. It cannot be directly written to. Instead, DBSR bits corresponding to 1 bits in the GPR can be cleared using **mtspr**.

3 Implementations may support more than one PID. For implementations with multiple PIDs, the PID defined by the embedded category is PID0.

4 User-mode read access to SPRG3 is implementation dependent.

5 The TSR is read using **mfspr**. It cannot be directly written to. Instead, TSR bits corresponding to 1 bits in the GPR can be cleared using **mtspr**.

6 USPRG0 is a separate physical register from SPRG0.

Table 6-2 describes the implementation-specific SPRs and SPRs defined by categories other than the base and embedded categories. Compilers should recognize the mnemonic names given in Table 6-2 when parsing instructions.

Table 6-2. Additional SPRs (by SPR Abbreviation)

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
BBEAR	Branch buffer entry address register [1]	513	Read/Write	No	6.9.1/6-23
BBTAR	Branch buffer target address register [1]	514	Read/Write	No	6.9.2/6-23
BUCSR	Branch unit control and status register [1]	1013	Read/Write	Yes	6.9.3/6-24
HID0	Hardware implementation dependent reg 0 [1]	1008	Read/Write	Yes	6.10.1/6-25
HID1	Hardware implementation dependent reg 1 [1]	1009	Read/Write	Yes	6.10.2/6-26
IVOR32	SPE unavailable interrupt offset	528	Read/Write	Yes	6.7.1.7/6-18
IVOR33	Floating-point data exception interrupt offset	529	Read/Write	Yes	
IVOR34	Floating-point round exception interrupt offset	530	Read/Write	Yes	
IVOR35	Performance monitor	531	Read/Write	Yes	
L1CFG0	L1 cache configuration register 0	515	Read only	No	6.11.3/6-30
L1CFG1	L1 cache configuration register 1	516	Read only	No	6.11.4/6-31
L1CSR0	L1 cache control and status register 0 [1]	1010	Read/Write	Yes	6.11.1/6-28
L1CSR1	L1 cache control and status register 1 [1]	1011	Read/Write	Yes	6.11.2/6-29
MAS0	MMU assist register 0 [1]	624	Read/Write	Yes	6.12.5.1/6-34
MAS1	MMU assist register 1 [1]	625	Read/Write	Yes	6.12.5.2/6-35
MAS2	MMU assist register 2 [1]	626	Read/Write	Yes	6.12.5.3/6-36
MAS3	MMU assist register 3 [1]	627	Read/Write	Yes	6.12.5.4/6-37
MAS4	MMU assist register 4 [1]	628	Read/Write	Yes	6.12.5.5/6-37
MAS6	MMU assist register 6 [1]	630	Read/Write	Yes	6.12.5.6/6-38
MCAR	Machine check address register	573	Read only	Yes	6.7.2.3/6-21
MCSR	Machine check syndrome register	572	Read/Write	Yes	6.7.2.4/6-21
MCSRR0	Machine check save/restore register 0	570	Read/Write	Yes	6.7.2.1/6-20
MCSRR1	Machine check save/restore register 1	571	Read/Write	Yes	6.7.2.2/6-20
MMUCFG	MMU configuration register	1015	Read only	Yes	6.12.3/6-32
MMUCSR0	MMU control and status register 0 [1]	1012	Read/Write	Yes	6.12.2/6-32
PID0	Process ID register 0 [1]	48	Read/Write	Yes	6.12.1/6-32
PID1	Process ID register 1 [1]	633	Read/Write	Yes	
PID2	Process ID register 2 [1]	634	Read/Write	Yes	
SPEFSCR	Signal processing and embedded floating-point status and control register [1]	512	Read/Write	No	6.14/6-44

Table 6-2. Additional SPRs (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
SVR	System version register	1023	Read only	Yes	6.5.4/6-14
TLB0CFG	TLB configuration register 0	688	Read only	Yes	6.12.4/6-33
TLB1CFG	TLB configuration register 1	689	Read only	Yes	6.12.4.2/6-34

[1] Accesses to this register requires synchronization, as described in the “Synchronization Requirements” section of the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*.

6.3 Registers for Computational Operations

The following sections describe general-purpose and integer exception registers.

NOTE

Register fields designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

6.3.1 General-Purpose Registers (GPRs)

GPR0–GPR31 support integer operations. The instruction formats provide 5-bit fields for specifying the GPRs to be used in the execution of the instruction. Each GPR is a 64-bit register, although only 64-bit load, store, and merge instructions use GPR bits 0–31.

6.3.2 Integer Exception Register (XER)

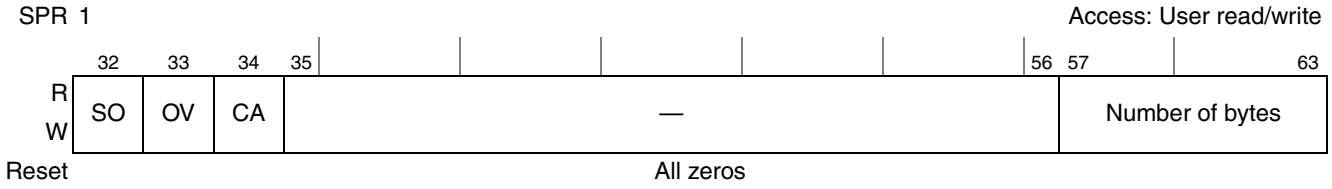


Figure 6-2. Integer Exception Register (XER)

Table 6-3. XER Field Description

Bits	Name	Description
32	SO	Summary overflow. Set when an instruction (except **mtspr**) sets the overflow bit. Once set, SO remains set until it is cleared by **mtspr[XER]** or **mcrxr**. SO is not altered by compare instructions or by other instructions (except **mtspr[XER]** and **mcrxr**) that cannot overflow. Executing **mtspr[XER]**, supplying the values 0 for SO and 1 for OV, causes SO to be cleared and OV to be set.
33	OV	Overflow. X-form add, subtract from, and negate instructions having OE = 1 set OV if the carry out of bit 32 is not equal to the carry out of bit 33, and clear OV otherwise to indicate a signed overflow. X-form multiply low word and divide word instructions having OE = 1 set OV if the result cannot be represented in 32 bits (**mullwo**, **divwo**, and **divwuo**) and clear OV otherwise. OV is not altered by compare instructions or by other instructions (except **mtspr[XER]** and **mcrxr**) that cannot overflow.

Table 6-3. XER Field Description (continued)

Bits	Name	Description
34	CA	Carry. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA if there is a carry out of bit 32 and clear it otherwise. CA can be used to indicate unsigned overflow for add and subtract operations that set CA. Shift right algebraic word instructions set CA if any 1 bits are shifted out of a negative operand and clear CA otherwise. Compare instructions and instructions that cannot carry (except Shift Right Algebraic Word, **mtspr[XER]**, and **mcrxr**) do not affect CA.
35–56	—	Reserved, should be cleared.
57–63	No. of bytes	Supports emulation of load and store string instructions. Specifies the number of bytes to be transferred by a load string indexed or store string indexed instruction.

6.4 Registers for Branch Operations

This section describes registers that support branch and CR operations.

6.4.1 Condition Register (CR)

Access: User read/write

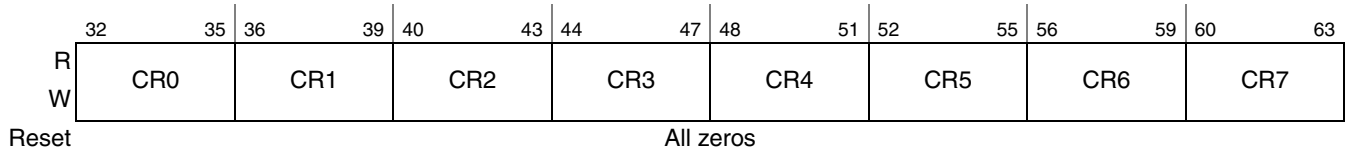


Figure 6-3. Condition Register (CR)

Table 6-4. BI Operand Settings for CR Fields

CR*n* Bits	CR Bits	BI	Description
CR0[0]	32	00000	Negative (LT)—Set when the result is negative. For SPE vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR0[1]	33	00001	Positive (GT)—Set when the result is positive (and not zero). For SPE vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.
CR0[2]	34	00010	Zero (EQ)—Set when the result is zero. For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR0[3]	35	00011	Summary overflow (SO). Copy of XER[SO] at the instruction's completion. For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.
CR1[0]	36	00100	Negative (LT) For SPE vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR1[1]	37	00101	Positive (GT) For SPE vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.

Table 6-4. BI Operand Settings for CR Fields (continued)

CR*n* Bits	CR Bits	BI	Description
CR1[2]	38	00110	Zero (EQ) For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR1[3]	39	00111	Summary overflow (SO) For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.
CR*n*[0]	40 44 48 52 56 60	01000 01100 10000 10100 11000 11100	Less than (LT) For integer compare instructions: **r**A < SIMM or rB (signed comparison) or **r**A < UIMM or **r**B (unsigned comparison). For SPE vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR*n*[1]	41 45 49 53 57 61	01001 01101 10001 10101 11001 11101	Greater than (GT) For integer compare instructions: **r**A > SIMM or **r**B (signed comparison) or **r**A > UIMM or **r**B (unsigned comparison). For SPE vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.
CR*n*[2]	42 46 50 54 58 62	01010 01110 10010 10110 11010 11110	Equal (EQ) For integer compare instructions: **r**A = SIMM, UIMM, or **r**B. For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR*n*[3]	43 47 51 55 59 63	01011 01111 10011 10111 11011 11111	Summary overflow (SO) For integer compare instructions, this is a copy of XER[SO] at the completion of the instruction. For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.

The bits of CR0 are interpreted as described in Table 6-5.

Table 6-5. CR0 Bit Descriptions

CR Bit	Name	Description
32	Negative (LT)	Bit 32 of the result is equal to 1.
33	Positive (GT)	Bit 32 of the result is equal to 0 and at least one bit from 33–63 of the result is non-zero.
34	Zero (EQ)	Bits 32–63 of the result are equal to 0.
35	Summary overflow (SO)	This is a copy of the final state of XER[SO] at the completion of the instruction.

6.4.2 Link Register (LR)

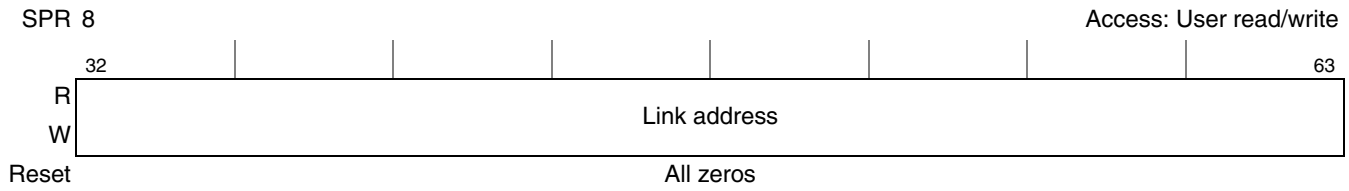


Figure 6-4. Link Register (LR)

6.4.3 Count Register (CTR)

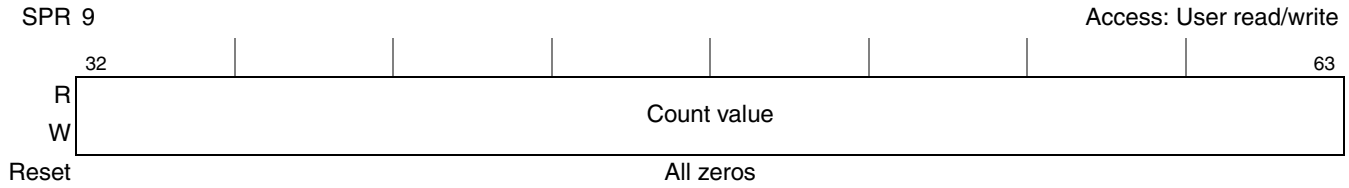


Figure 6-5. Count Register (CTR)

6.5 Processor Control Registers

This section addresses machine state, processor ID, and processor version registers.

6.5.1 Machine State Register (MSR)

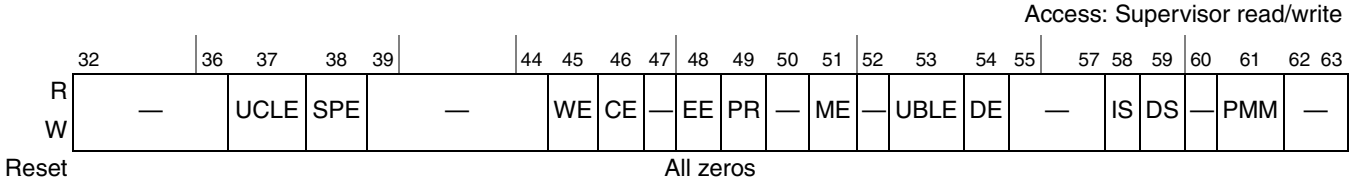


Figure 6-6. Machine State Register (MSR)

Table 6-6. MSR Field Descriptions

Bits	Name	Description
32–36	—	Reserved, should be cleared.[1]
37	UCLE	User-mode cache lock enable. Used to restrict user-mode cache-line locking by the operating system 0 Any cache lock instruction executed in user-mode takes a cache-locking DSI exception and sets either ESR[DLK] or ESR[ILK]. This allows the operating system to manage and track the locking/unlocking of cache lines by user-mode tasks. 1 Cache-locking instructions can be executed in user-mode and they do not take a DSI for cache-locking (they may still take a DSI for access violations though).

Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
38	SPE	SPE enable. (e500-specific). 0 If software attempts to execute an instruction that accesses the upper word of a GPR, the SPE unavailable exception is taken. The exception is also taken if software attempts to execute scalar floating-point instructions. 1 Software can execute the following instructions: These instructions include the SPE instructions and both vector and scalar single-precision floating-point instructions.
39–44	—	Reserved, should be cleared. [1]
45	WE	Wait state enable. Allows the core complex to signal a request for power management, according to the states of HID0[DOZE], HID0[NAP], and HID0[SLEEP]. 0 The processor is not in wait state and continues processing. No power management request is signaled to external logic. 1 The processor enters wait state by ceasing to execute instructions and entering low-power mode. Details of how wait state is entered and exited and how the processor behaves in the wait state are implementation-dependent. On the e500, MSR[WE] gates the DOZE, NAP, and SLEEP outputs from the core complex; as a result, these outputs negate to the external power management logic on entry to the interrupt and then return to their previous state on return from the interrupt. WE is cleared on entry to any interrupt and restored to its previous state upon return.
46	CE	Critical enable 0 Critical input and watchdog timer interrupts are disabled. 1 Critical input and watchdog timer interrupts are enabled.
47	—	Reserved, should be cleared. [1]
48	EE	External enable 0 External input, decrementer, fixed-interval timer, and performance monitor interrupts are disabled. 1 External input, decrementer, fixed-interval timer, and performance monitor interrupts are enabled.
49	PR	User mode (problem state) 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (for example, GPRs, SPRs, and the MSR). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource. PR also affects memory access control
50	—	Reserved, should be cleared. [1]
51	ME	Machine check enable 0 Machine check interrupts are disabled. 1 Machine check interrupts are enabled.
52	—	Reserved, should be cleared. [1]
53	UBLE	In the e500, it is the user BTB lock enable bit. 0 User-mode execution of the BTB lock instructions is disabled; privileged instruction exception taken instead. 1 User-mode execution of the BTB lock instructions for user mode is enabled.
54	DE	Debug interrupt enable. See the description of the DBSR[UDE] in Section 6.13.2, “Debug Status Register (DBSR).” 0 Debug interrupts are disabled. 1 Debug interrupts are enabled if DBCR0[IDM] = 1.
55–57	—	Reserved, should be cleared. [1]

Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
58	IS	Instruction address space 0 The processor directs all instruction fetches to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs all instruction fetches to address space 1 (TS = 1 in the relevant TLB entry).
59	DS	Data address space 0 The processor directs data memory accesses to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs data memory accesses to address space 1 (TS = 1 in the relevant TLB entry).
60	—	Reserved, should be cleared. [1]
61	PMM	Performance monitor mark bit. System software can set PMM when a marked process is running to enable statistics to be gathered only during execution of the marked process. MSR[PR] and MSR[PMM] together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCax, the state for which monitoring is enabled, counting is enabled.
62–63	—	Preserved for OEA-defined RI and LE, respectively

[1] An MSR bit that is reserved may be altered by a return from interrupt instruction.

6.5.2 Processor ID Register (PIR)

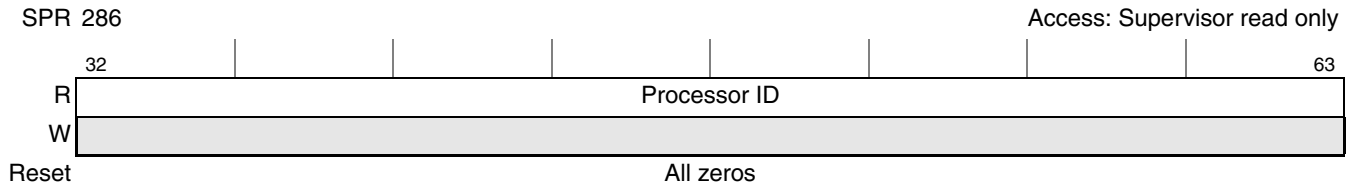


Figure 6-7. Processor ID Register (PIR)

6.5.3 Processor Version Register (PVR)

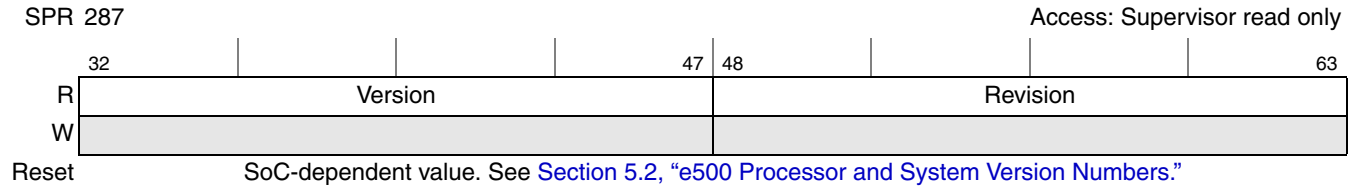


Figure 6-8. Processor Version Register (PVR)

Table 6-7. PVR Field Descriptions

Bits	Name	Description
32–47	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported. (See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.)
48–63	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level. (See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.)

6.5.4 System Version Register (SVR)

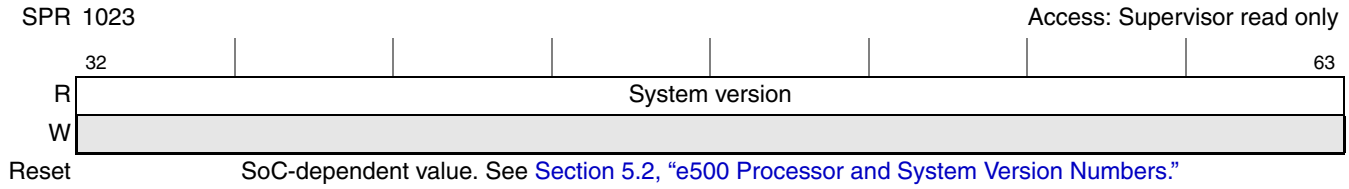


Figure 6-9. System Version Register (SVR)

Table 6-8. SVR Field Descriptions

Bits	Name	Description
32–63	System version	A 16-bit number that identifies the SoC version. See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.

6.6 Timer Registers

6.6.1 Timer Control Register (TCR)

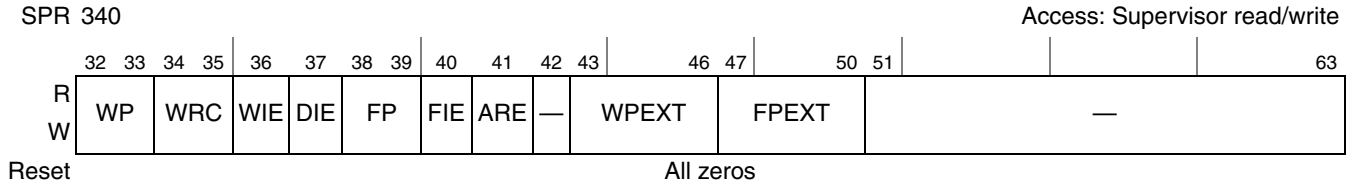


Figure 6-10. Timer Control Register (TCR)

Table 6-9. TCR Field Descriptions

Bits	Name	Description
32–33	WP	Watchdog timer period. When concatenated with WPEXT, specifies one of 64-bit locations of the time base used to signal a watchdog timer exception on a transition from 0 to 1. WPEXT[0–3] || WP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) WPEXT[0–3] || WP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
34–35	WRC	Watchdog timer reset control. This value is written into TSR[WRS] when a watchdog event occurs. WRC may be set by software but cannot be cleared by software, except by a software-induced reset. Once written to a non-zero value, WRC may no longer be altered by software. 00 No watchdog timer reset will occur. 01 If MSR[ME] = 0, the second timeout is ignored. If MSR[ME] = 1, a machine check condition occurs on a second timeout of the watchdog timer, and if HID0[EMCP] = 1, the machine check interrupt is generated. 10 Assert processor reset output (*core_hreset_req*) on second timeout of watchdog timer 11 Reserved
36	WIE	Watchdog timer interrupt enable 0 Watchdog timer interrupts disabled 1 Watchdog timer interrupts enabled
37	DIE	Decrementer interrupt enable 0 Decrementer interrupts disabled 1 Decrementer interrupts enabled
38–39	FP	Fixed interval timer period. When concatenated with FPEXT, FP specifies one of 64 bit locations of the time base used to signal a fixed-interval timer exception on a transition from 0 to 1. FPEXT[0–3] || FP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) FPEXT[0–3] || FP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
40	FIE	Fixed interval interrupt enable 0 Fixed interval interrupts disabled 1 Fixed interval interrupts enabled
41	ARE	Auto-reload enable. Controls whether the DECAR value is reloaded into the DEC when the DEC value reaches 0000_0001. See *EREF: A Reference for Freescale Book E and the e500 Core*. 0 Auto-reload disabled 1 Auto-reload enabled
42	—	Reserved, should be cleared.
43–46	WPEXT	Watchdog timer period extension (see the description for WP)
47–50	FPEXT	Fixed-interval timer period extension (see the description for FP)
51–63	—	Reserved, should be cleared.

6.6.2 Timer Status Register (TSR)

SPR 336 — Access: Supervisor w1c[1]

	32	33	34 35	36	37	38 63
R	ENW	WIS	WRS	DIS	FIS	—
W	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros					

[1] Set by hardware. Read with **mfspr** and cleared with **mtspr** by writing ones to any TSR bit positions to be cleared and zeros in all other bit positions.

Figure 6-11. Timer Status Register (TSR)

Table 6-10. TSR Field Descriptions

Bits	Name	Description
32	ENW	Enable next watchdog time. Functions as write-one-to-clear. 0 Action on next watchdog timer time-out is to set TSR[ENW] 1 Action on next watchdog timer time-out is governed by TSR[WIS] When a watchdog timer time-out occurs while WIS = 0 and the next watchdog time-out is enabled (ENW = 1), a watchdog timer exception is generated and logged by setting WIS. This is referred to as a watchdog timer first time out. A watchdog timer interrupt occurs if enabled by TCR[WIE] and MSR[CE]. To avoid another watchdog timer interrupt once MSR[CE] is reenabled, (assuming TCR[WIE] is not cleared instead), the interrupt handler must reset TSR[WIS].
33	WIS	Watchdog timer interrupt status. Functions as write-one-to-clear. 0 A watchdog timer event has not occurred. 1 A watchdog timer event occurred. When MSR[CE] = 1 and TCR[WIE] = 1, a watchdog timer interrupt is taken. See the description of ENW for more information about how WIS is used.
34–35	WRS	Watchdog timer reset status. Functions as write-one-to-clear. Defined at reset (value = 00). Set to TCR[WRC] when a reset is caused by the watchdog timer.
36	DIS	Decrementer interrupt status. Functions as write-one-to-clear. 0 A decrementer event has not occurred. 1 A decrementer event occurred. When MSR[EE] = TCR[DIE] = 1, a decrementer interrupt is taken.
37	FIS	Fixed-interval timer interrupt status. Functions as write-one-to-clear. 0 A fixed-interval timer event has not occurred. 1 A fixed-interval timer event occurred. When MSR[EE] = 1 and TCR[FIE] = 1, a fixed-interval timer interrupt is taken.
38–63	—	Reserved, should be cleared.

6.6.3 Time Base Registers

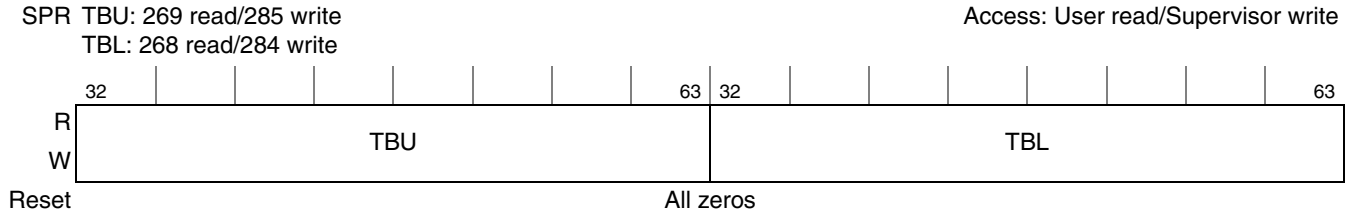


Figure 6-12. Time Base Upper/Lower Registers (TBU/TBL)

6.6.4 Decrementer Register

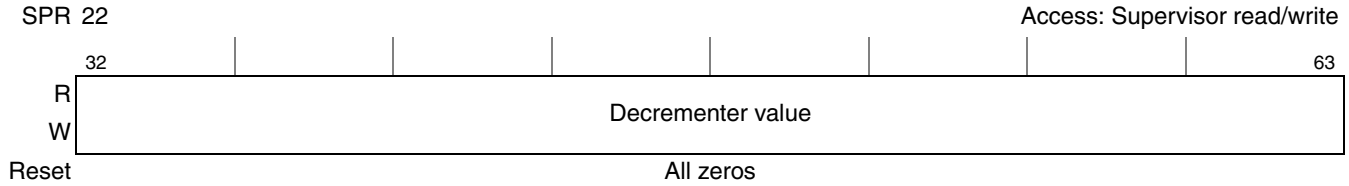


Figure 6-13. Decrementer Register (DEC)

6.6.5 Decrementer Auto-Reload Register (DECAR)

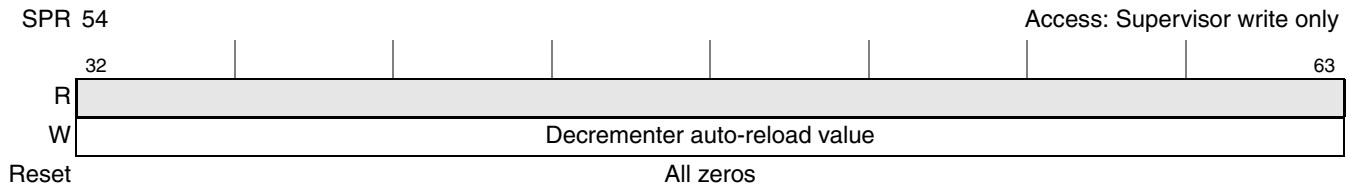


Figure 6-14. Decrementer Auto-Reload Register (DECAR)

6.7 Interrupt Registers

6.7.1 Interrupt Registers Defined by the Embedded and Base Categories

6.7.1.1 Save/Restore Register 0 (SRR0)

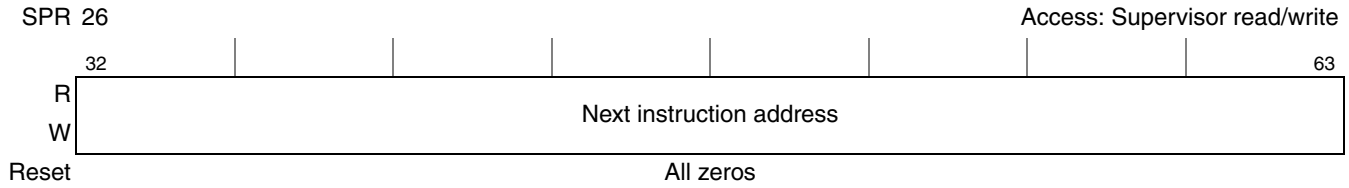


Figure 6-15. Save/Restore Register 0 (SRR0)

6.7.1.2 Save/Restore Register 1 (SRR1)

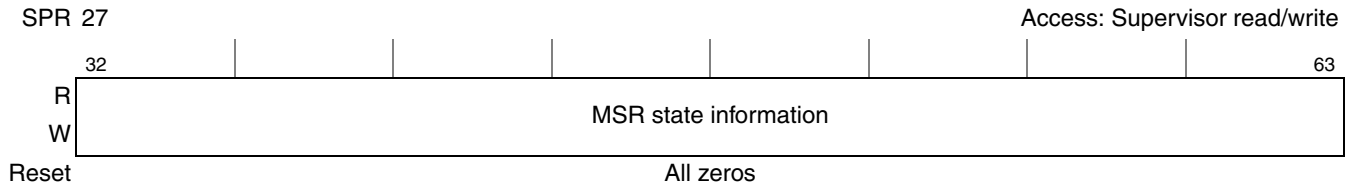


Figure 6-16. Save/Restore Register 1 (SRR1)

6.7.1.3 Critical Save/Restore Register 0 (CSRR0)

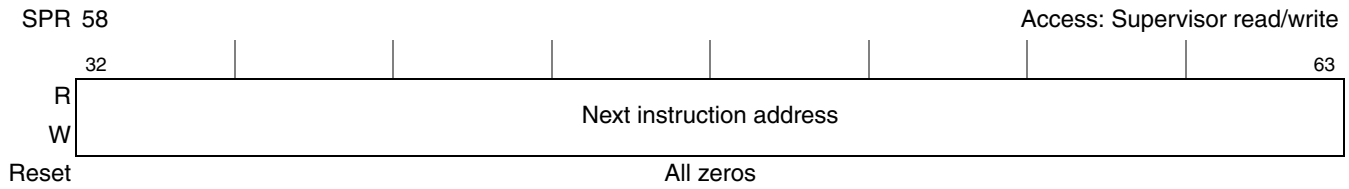


Figure 6-17. Critical Save/Restore Register 0 (CSRR0)

6.7.1.4 Critical Save/Restore Register 1 (CSRR1)

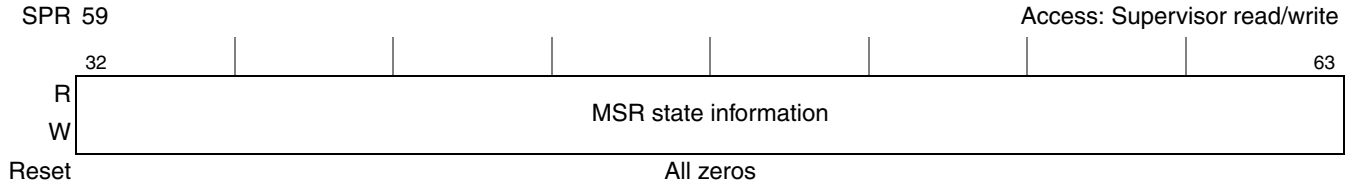


Figure 6-18. Critical Save/Restore Register 1 (CSRR1)

6.7.1.5 Data Exception Address Register (DEAR)

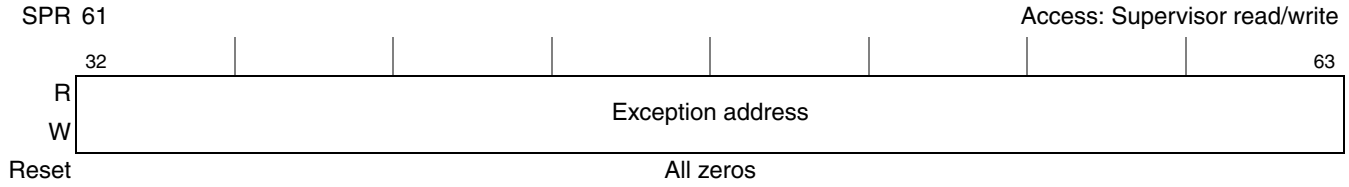


Figure 6-19. Data Exception Address Register (DEAR)

6.7.1.6 Interrupt Vector Prefix Register (IVPR)

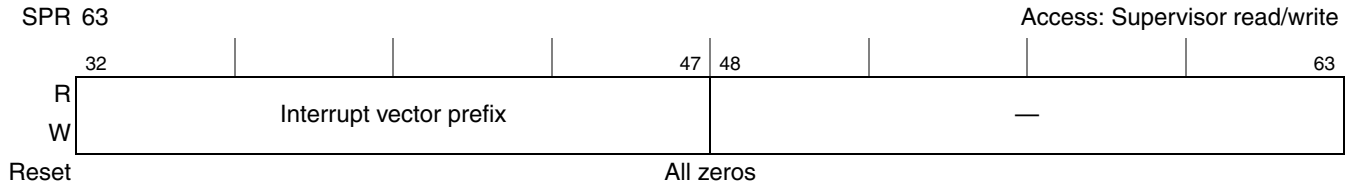


Figure 6-20. Interrupt Vector Prefix Register (IVPR)

6.7.1.7 Interrupt Vector Offset Registers (IVOR*n*)

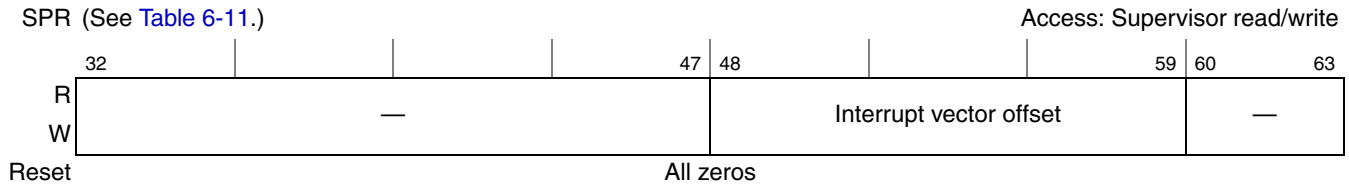


Figure 6-21. Interrupt Vector Offset Registers (IVOR*n*)

Table 6-11. IVOR Assignments

IVOR Number	SPR	Interrupt Type
IVOR0	400	Critical input
IVOR1	401	Machine check
IVOR2	402	Data storage
IVOR3	403	Instruction storage
IVOR4	404	External input

Table 6-11. IVOR Assignments (continued)

IVOR Number	SPR	Interrupt Type
IVOR5	405	Alignment
IVOR6	406	Program
IVOR8	408	System call
IVOR10	410	Decrementer
IVOR11	411	Fixed-interval timer interrupt
IVOR12	412	Watchdog timer interrupt
IVOR13	413	Data TLB error
IVOR14	414	Instruction TLB error
IVOR15	415	Debug
IVOR16–IVOR31	—	Reserved for future architectural use
IVOR32	528	SPE unavailable
IVOR33	529	Floating-point data exception
IVOR34	530	Floating-point round exception
IVOR35	531	Performance monitor
IVOR36–IVOR63	—	Allocated for implementation-dependent use

6.7.1.8 Exception Syndrome Register (ESR)

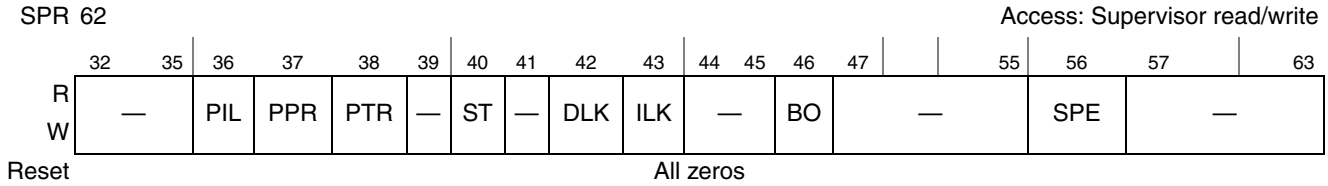


Figure 6-22. Exception Syndrome Register (ESR)

Table 6-12. ESR Field Descriptions

Bits	Name	Syndrome	Interrupt Types
32–35	—	Reserved, should be cleared.	—
36	PIL	Illegal instruction exception	Program
37	PPR	Privileged instruction exception	Program
38	PTR	Trap exception	Program
39	—	Reserved and permanently cleared because the e500 does not implement this FPU. Setting it has no effect.	—
40	ST	Store operation	Alignment, data storage, data TLB error

Table 6-12. ESR Field Descriptions (continued)

Bits	Name	Syndrome	Interrupt Types
41	—	Reserved, should be cleared.	—
42	DLK	Cache locking. Settings are implementation-dependent. 0 Default 1 On the e500, DLK is set when a DSI occurs because **dcbtls**, **dcbtstls**, or **dcblc** is executed in user mode while MSR[UCLE] = 0.	Data storage
43	ILK	Set when a DSI occurs because **icbtl** or **icblc** is executed in user mode (MSR[PR] = 1) and MSR[UCLE] = 0	Data storage
44–45	—	Reserved, should be cleared.	
46	BO	Byte-ordering exception	Data storage, instruction storage
47–55	—	Reserved, should be cleared.	—
56	SPE	SPE exception bit (e500-specific) 0 Default 1 Any exception caused by an SPE or SPFP instruction	SPE unavailable
57–63	—	Reserved, should be cleared.	—

6.7.2 Additional Interrupt Registers

6.7.2.1 Machine Check Save/Restore Register 0 (MCSRR0)

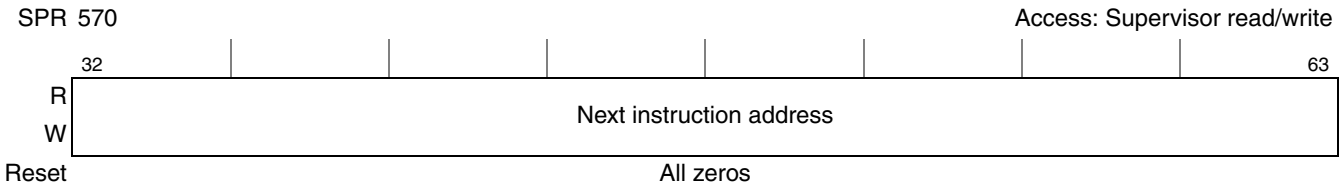


Figure 6-23. Machine Check Save/Restore Register 0 (MCSRR0)

6.7.2.2 Machine Check Save/Restore Register 1 (MCSRR1)

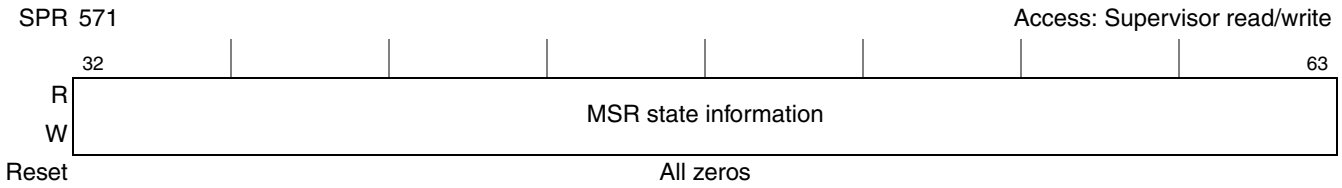


Figure 6-24. Machine Check Save/Restore Register 1 (MCSRR1)

6.7.2.3 Machine Check Address Register (MCAR)

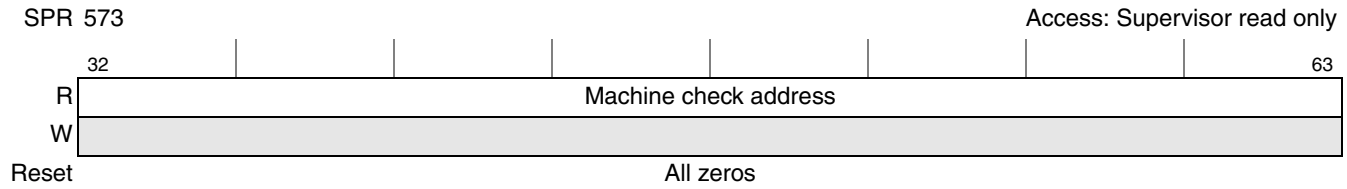


Figure 6-25. Machine Check Address Register (MCAR)

6.7.2.4 Machine Check Syndrome Register (MCSR)

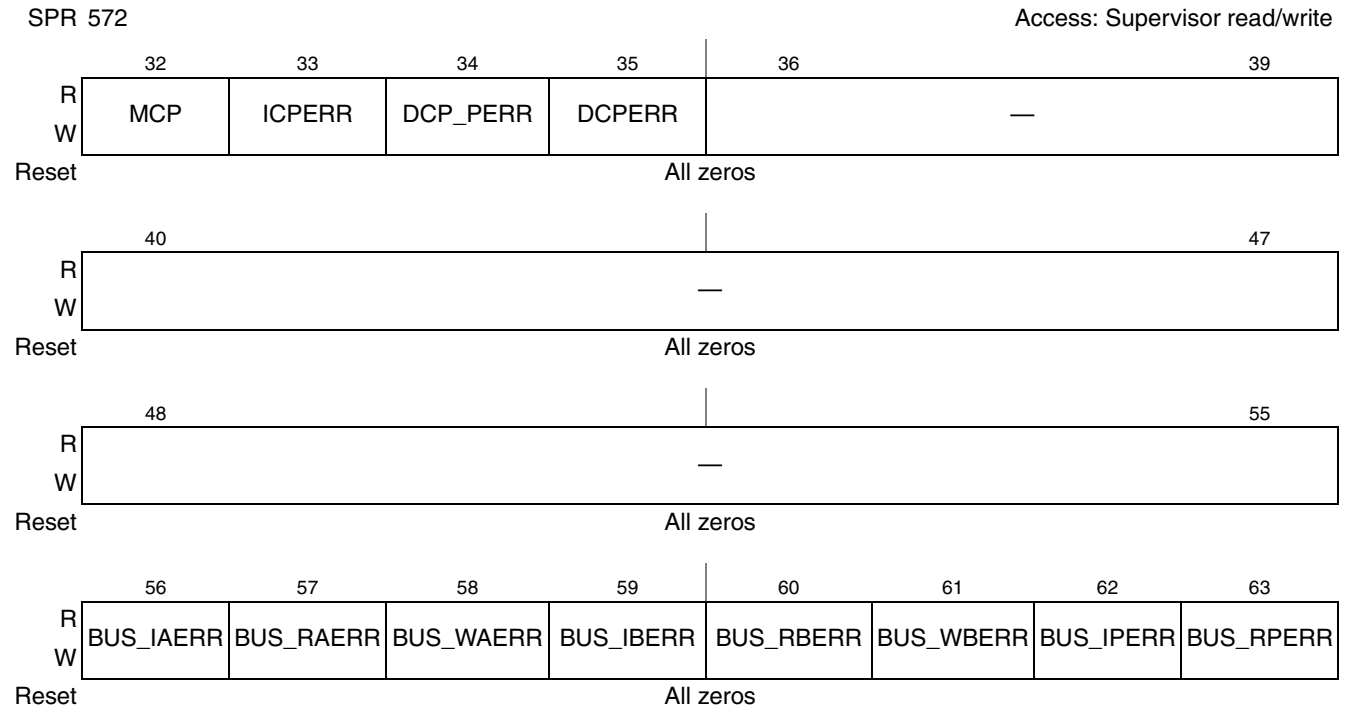


Figure 6-26. Machine Check Syndrome Register (MCSR)

Table 6-13. MCSR Field Descriptions

Bits	Name	Description
32	MCP	Machine check input pin
33	ICPERR	Instruction cache parity error
34	DCP_PERR	Data cache push parity error
35	DCPERR	Data cache parity error
36–55	—	Reserved, should be cleared.
56	BUS_IAERR	Bus instruction address error
57	BUS_RAERR	Bus read address error
58	BUS_WAERR	Bus write address error

Table 6-13. MCSR Field Descriptions (continued)

Bits	Name	Description
59	BUS_IBERR	Bus instruction data bus error
60	BUS_RBERR	Bus read data bus error
61	BUS_WBERR	Bus write bus error
62	BUS_IPERR	Bus instruction parity error
63	BUS_RPERR	Bus read parity error

6.8 Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

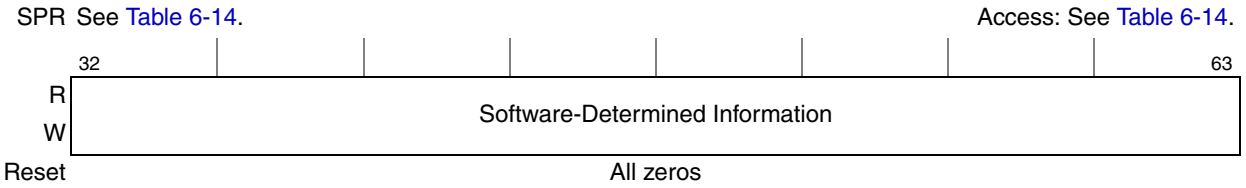


Figure 6-27. Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

Table 6-14. SPR Assignments

Name	SPR	Access	Level
SPRG0	272	Read/Write	Supervisor
SPRG1	273	Read/Write	Supervisor
SPRG2	274	Read/Write	Supervisor
SPRG3	259	Read only	User/Supervisor
	275	Read/Write	Supervisor
SPRG4	260	Read only	User/Supervisor
	276	Read/Write	Supervisor
SPRG5	261	Read only	User/Supervisor
	277	Read/Write	Supervisor
SPRG6	262	Read only	User/Supervisor
	278	Read/Write	Supervisor
SPRG7	263	Read only	User/Supervisor
	279	Read/Write	Supervisor
USPRG0	256	Read/Write	User/Supervisor

6.9 Branch Target Buffer (BTB) Registers

6.9.1 Branch Buffer Entry Address Register (BBEAR)

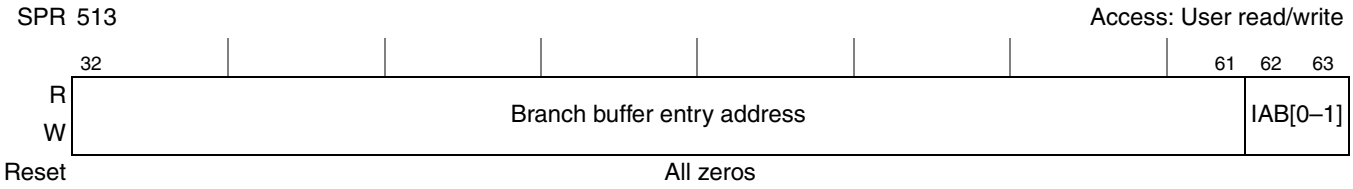


Figure 6-28. Branch Buffer Entry Address Register (BBEAR)

Table 6-15. BBEAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer entry address	Branch buffer entry effective address bits 0–29
62–63	IAB[0–1]	Instruction after branch (with BBTAR[62]). 3-bit pointer that points to the instruction in the cache line after the branch. See the description in the **bblels** instruction in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.

6.9.2 Branch Buffer Target Address Register (BBTAR)

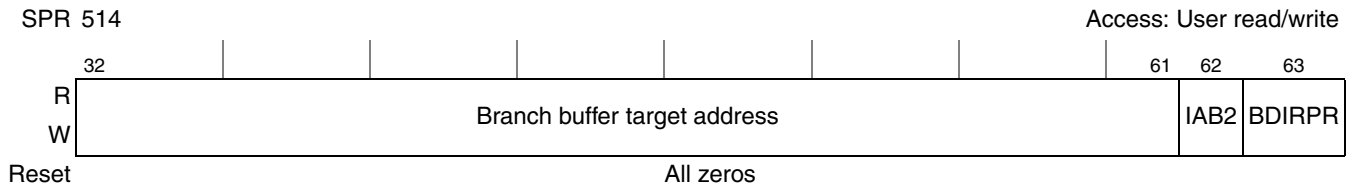


Figure 6-29. Branch Buffer Target Address Register (BBTAR)

Table 6-16. BBTAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer target address	Branch buffer target effective address bits 0–29
62	IAB2	Instruction after branch bit 2 (with BBEAR[62–63]). IAB is a 3-bit pointer that points to the instruction in the cache line after the branch. See the description for **bblels** in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.
63	BDIRPR	Branch direction prediction. The user can pick the direction of the predicted branch. 0 The locked address is always predicted as not taken. 1 The locked address is always predicted as taken.

6.9.3 Branch Unit Control and Status Register (BUCSR)

SPR 1013 Access: Supervisor read/write

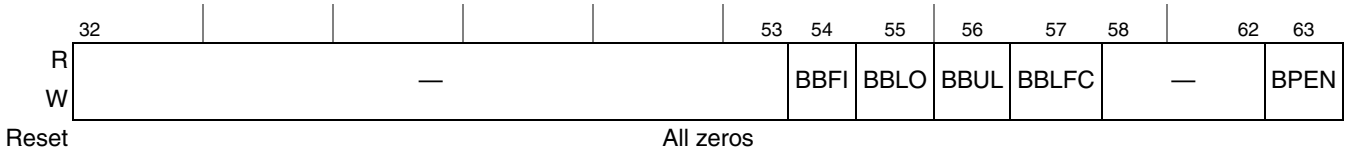


Figure 6-30. Branch Unit Control and Status Register (BUCSR)

Table 6-17. BUCSR Field Descriptions

Bits	Name	Description
32–53	—	Reserved, should be cleared.
54	BBFI	Branch buffer flash invalidate. Clearing and then setting BBFI flash clears the valid bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBFI is always read as 0.
55	BBLO	Branch buffer lock overflow status 0 Indicates a lock overflow condition was not encountered in the branch buffer 1 Indicates a lock overflow condition was encountered in the branch buffer This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
56	BBUL	Branch buffer unable to lock 0 Indicates a lock overflow condition in the branch buffer 1 Indicates a lock set instruction failed in the branch buffer, for example, if the BTB is disabled. This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
57	BBLFC	Branch buffer lock bits flash clear. Clearing and then setting BBLFC flash clears the lock bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBLFC is always read as 0.
58–62	—	Reserved, should be cleared.
63	BPEN	Branch prediction enable 0 Branch prediction disabled 1 Branch prediction enabled (enables BTB to predict branches)

6.10 Hardware Implementation-Dependent Registers

6.10.1 Hardware Implementation-Dependent Register 0 (HID0)

SPR 1008 Access: Supervisor read/write

	32	33–39	40	41	42	43–47
R W	EMCP	—	DOZE	NAP	SLEEP	—
Reset	All zeros					

	48	49	50	51–62	63
R W	—	TBEN	SEL_TBCLK	—	NOPTI
Reset	All zeros				

Figure 6-31. Hardware Implementation-Dependent Register 0 (HID0)

Table 6-18. HID0 Field Descriptions

Bits	Name	Description
32	EMCP	Enable machine check pin, $\overline{\text{MCP}}$. Used to mask out further machine check exceptions caused by assertion of $\overline{\text{MCP}}$. 0 $\overline{\text{MCP}}$ is disabled. 1 $\overline{\text{MCP}}$ is enabled. If MSE[ME] = 0, asserting $\overline{\text{MCP}}$ causes checkstop. If MSR[ME] = 1, asserting $\overline{\text{MCP}}$ causes a machine check exception.
33–39	—	Reserved, should be cleared.
40	DOZE	Doze power management mode. If MSR[WE] is set, this bit controls DOZE mode. 0 Core not in doze mode 1 Core in doze mode
41	NAP	Nap power management mode. If MSR[WE] is set, this bit controls NAP mode. 0 Core not in nap mode 1 Core in nap mode
42	SLEEP	Configure for sleep power management mode. Controls SLEEP mode if MSR[WE] is set. 0 Core not in sleep mode 1 Core in sleep mode
43–48	—	Reserved, should be cleared.
49	TBEN	Time base enable 0 Time base disabled (no counting) 1 Time base enabled • If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 0, the time base is updated every 8 bus clocks • If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 1, the time base is updated on the rising edge of *core_tbclk* (sampled at bus rate). The maximum supported frequency can be found in the electrical specifications, but this value is approximately 25% of the bus clock frequency.
50	SEL_TBCLK	Select time base clock. If the time base is enabled, this field functions as follows: 0 Time base is based on the processor clock 1 Time base is based on the TBCLK (RTC) input

Table 6-18. HID0 Field Descriptions (continued)

Bits	Name	Description
51–62	—	Reserved, should be cleared.
63	NOPTI	No-op the data and instruction cache touch instructions. 0 **dcbt**, **dcbtst**, and **icbt** are enabled. On the e500, if CT = 0, **icbt** is always a no-op, regardless of the value of NOPTI. If CT = 1, **icbt** does a touch load to an L2 cache, if one is present. 1 **dcbt**, **dcbtst**, and **icbt** are treated as no-ops; **dcblc** and **dcbtls** are not.

6.10.2 Hardware Implementation-Dependent Register 1 (HID1)

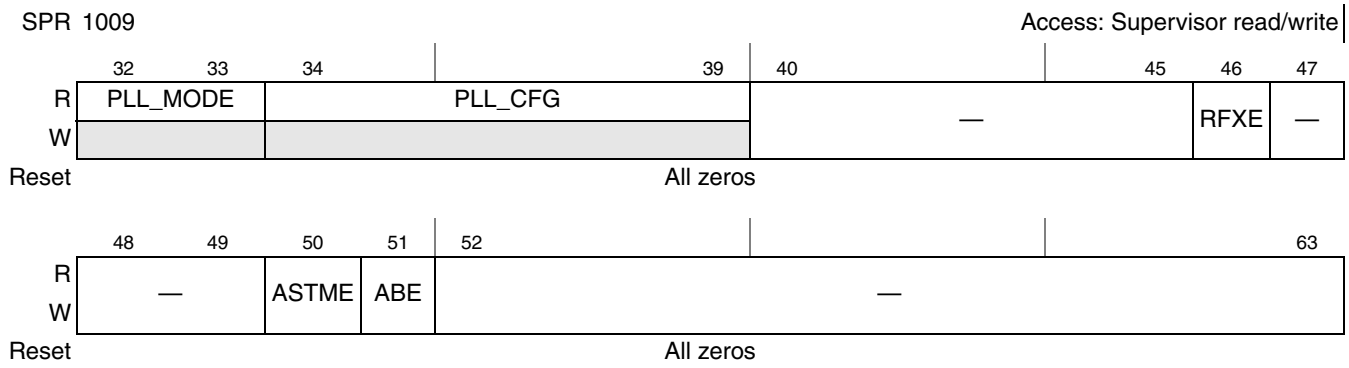


Figure 6-32. Hardware Implementation-Dependent Register 1 (HID1)

Table 6-19. HID1 Field Descriptions

Bits	Name	Description
32–33	PLL_MODE	Read-only for integrated devices. 01 Fixed value for MPC8555E
34–39	PLL_CFG	Reflected directly from configuration input pins (read-only). PLL_CFG[0–4] corresponds to the integer divide ratio and PLL_CFG5 is the half-mode bit. The following values are supported: 0001_00 Ratio of 2:1 0001_01 Ratio of 5:2 (2.5:1) 0001_10 Ratio of 3:1 0001_11 Ratio of 7:2 (3.5:1) Note that this value is also reflected to PORPLLSR[e500_Ratio]. See Section 18.4.1.1, “POR PLL Status Register (PORPLLSR).”
40–45	—	Reserved, should be cleared.

Table 6-19. HID1 Field Descriptions (continued)

Bits	Name	Description
46	RFXE	Read fault exception enable. Enables the core to internally generate a machine check interrupt when *core_fault_in* is asserted. Depending on the value of MSR[ME], this results in either a machine check interrupt or a checkstop. 0 Assertion of *core_fault_in* cannot cause a machine check. The e500 does not stall when faulty instructions or data are received, as described in the following note. **Note:** The e500 does not stall when faulty instructions or data are received. Instead, it continues processing with faulty instructions or data. The only reliable way to prevent such behavior is to set RFXE, which causes a machine check before the faulty instructions or data are used. To avoid the use of faulty instructions or data and to have good error determination, software must set RFXE and program the PIC to interrupt the processor when errors occur. As a result, software must deal with multiple interrupts for the same fundamental problem. 1 Assertion of *core_fault_in* causes a machine check if MSR[ME] = 1 or a checkstop if MSR[ME] = 0. The *core_fault_in* signal is asserted to the core when logic outside of the core has a problem delivering good data to the core. For example, the front-side L2 cache asserts *core_fault_in* when an ECC error occurs and ECC is enabled. As a second example, it is asserted when there is a master abort on a PCI transaction. See “Proper Reporting of Bus Faults” in the core complex bus chapter of the *PowerPC™ e500 Core Family Reference Manual*. The RFXE bit provides flexibility in error recovery. Typically, devices outside the core have some way other than the assertion of *core_fault_in* to signal the core that an error occurred. Usually, this is done by channeling interrupt requests through a programmable interrupt controller (PIC) to the core. In these cases, the assertion of *core_fault_in* is used only to prevent the core from using bad data before receiving an interrupt from the PIC (for example, an external or critical input interrupt). Possible combinations of RFXE and PIC configuration are as follows: • RFXE = 0 and the PIC is configured to interrupt the processor. In this configuration, the assertion of *core_fault_in* does not trigger a machine check interrupt. However, there is a possibility that the core might use faulty data or instructions. The PIC interrupts the core so that error recovery can begin. This configuration allows the core to query the PIC and the rest of the system for more information about the cause of the interrupt, and generally provides the best error recovery capabilities. • RFXE = 1 and the PIC is not configured to interrupt the processor. This configuration provides quick error detection without the overhead of configuring the PIC. When the PIC is not configured, setting RFXE avoids stalling the core when *core_fault_in* is asserted. Determination of the root cause of the problem may be somewhat more difficult than it would be if the PIC were enabled. • RFXE = 1 and the PIC is configured to interrupt the processor. In this configuration, the core may receive two interrupts for the same fundamental error. The two interrupts may occur in any order, which may complicate error handling. Therefore, this is usually not an interesting configuration for a single-core device. This may, however, be an interesting configuration for multi-core devices in which the PIC may steer interrupts to a processor other than the one that attempted to fetch the faulty data. • RFXE = 0 and the PIC is not configured to interrupt the processor. This is not a recommended configuration. The processor may stall indefinitely due to an unreported error.
47–49	—	Reserved, should be cleared.
50	ASTME	Address bus streaming mode enable. This bit, along with the ECM stream control bits in the EEBACR, enables address bus streaming on the CCB. See Section 8.2.1.1, “ECM CCB Address Configuration Register (EEBACR).” 0 Address bus streaming mode disabled 1 Address bus streaming mode enabled

Table 6-19. HID1 Field Descriptions (continued)

Bits	Name	Description
51	ABE	Address broadcast enable. The e500 broadcasts cache management instructions (**dcbst**, **dcblc** (CT = 1), **icblc** (CT = 1), **dcbf**, **dcbi**, **mbar**, **msync**, **tlbsync**, **icbi**) based on ABE. ABE must be set to allow management of external L2 caches. 0 Address broadcasting disabled 1 Address broadcasting enabled
52–63	—	Reserved, should be cleared.

6.11 L1 Cache Configuration Registers

6.11.1 L1 Cache Control and Status Register 0 (L1CSR0)

SPR 1010 Access: Supervisor read/write

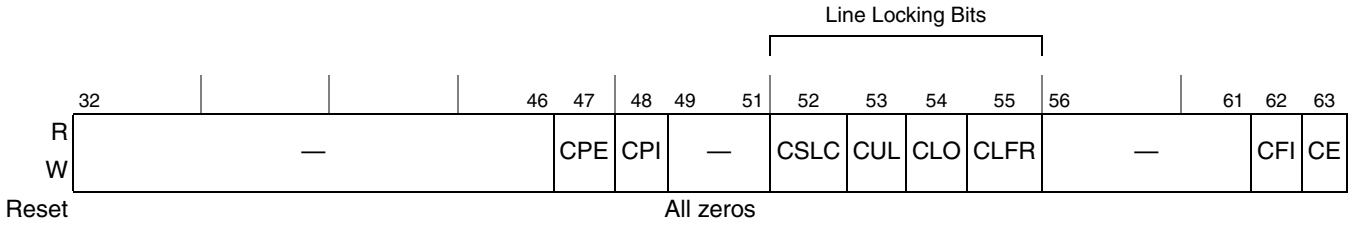


Figure 6-33. L1 Cache Control and Status Register 0 (L1CSR0)

Table 6-20. L1CSR0 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	CPE	(Data) Cache parity enable 0 Parity checking of the cache disabled 1 Parity checking of the cache enabled
48	CPI	(Data) Parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Cache parity must also be enabled (CPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
52	CSLC	(Data) Cache snoop lock clear. Sticky bit set by hardware if a **dcbi** snoop (either internally or externally generated) invalidated a locked cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can be cleared only by software. 0 The cache has not encountered a **dcbi** snoop that invalidated a locked line. 1 The cache has encountered a **dcbi** snoop that invalidated a locked line.
53	CUL	(Data) Cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the cache 1 Indicates a lock set instruction was not effective in the cache
54	CLO	(Data) Cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the cache 1 Indicates a lock overflow condition was encountered in the cache

Table 6-20. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
55	CLFR	(Data) Cache lock bits flash reset. Writing a 1 during a flash clear operation causes an undefined operation. Writing a 0 during a flash clear operation is ignored. Clearing occurs regardless of the enable (CE) value. 0 Default. 1 Hardware initiates a cache lock bits flash clear operation. This bit is cleared when the operation is complete.
56–61	—	Reserved, should be cleared.
62	CFI	(Data) Cache flash invalidate. 0 No cache invalidate. Writing a 0 to CFI during an invalidation operation is ignored. 1 Cache invalidation operation. A cache invalidation operation is initiated by hardware. Once complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation. Invalidation occurs regardless of the enable (CE) value.
63	CE	(Data) Cache enable 0 The cache is neither accessed or updated. 1 Enables cache operation

6.11.2 L1 Cache Control and Status Register 1 (L1CSR1)

SPR 1011 Access: Supervisor read/write

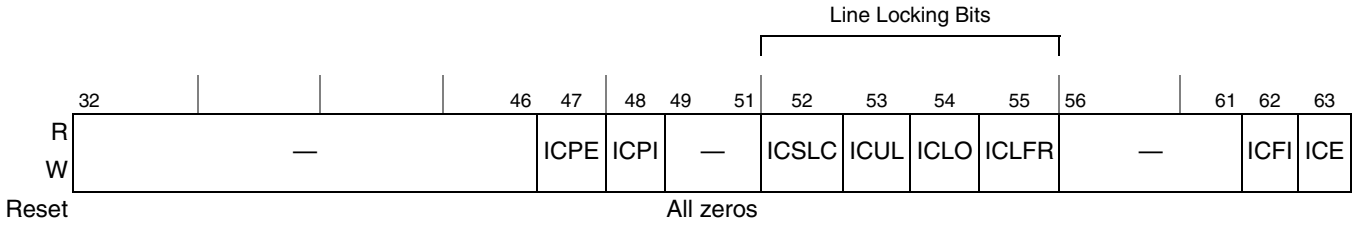


Figure 6-34. L1 Cache Control and Status Register 1 (L1CSR1)

Table 6-21. L1CSR1 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	ICPE	Instruction cache parity enable 0 Parity checking of the instruction cache disabled 1 Parity checking of the instruction cache enabled
48	ICPI	Instruction parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Note that instruction cache parity must also be enabled (ICPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
52	ICSLC	Instruction cache snoop lock clear. Sticky bit set by hardware if an **icbi** snoop (either internally or externally generated) invalidated a locked line in the instruction cache. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can only be cleared by software. 0 The instruction cache has not encountered an **icbi** snoop that invalidated a locked line. 1 The instruction cache has encountered an **icbi** snoop that invalidated a locked line.

Table 6-21. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
53	ICUL	Instruction cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the instruction cache 1 Indicates a lock set instruction was not effective in the instruction cache
54	ICLO	Instruction cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the instruction cache 1 Indicates a lock overflow condition was encountered in the instruction cache
55	ICLFR	Instruction cache lock bits flash reset. Writing 0 and then 1 flash clears the lock bit of all entries in the instruction cache; clearing occurs independently from the value of the enable bit (ICE). ICLFR is always read as 0.
56–61	—	Reserved, should be cleared.
62	ICFI	Instruction cache flash invalidate. Written to 0 and then 1 to flash clear the valid bit of all entries in the instruction cache; operates independently from the value of the enable bit (ICE). ICFI is always read as 0.
63	ICE	Instruction cache enable 0 The instruction cache is neither accessed or updated. 1 Enables instruction cache operation

6.11.3 L1 Cache Configuration Register 0 (L1CFG0)

SPR 515 Access: User read only

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	CARCH		—					CBSIZE		CREPL		CLA	CPA	—					CNWAY		—				CSIZE							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0

Figure 6-35. L1 Cache Configuration Register 0 (L1CFG0)

Table 6-22. L1CFG0 Field Descriptions

Bits	Name	Description
32–33	CARCH	Cache architecture 00 Harvard 01 Unified
34–38	—	Reserved, should be cleared.
39–40	CBSIZE	Cache line size 0 32 bytes 1 64 bytes
41–42	CREPL	Cache replacement policy 0 True LRU 1 Pseudo LRU
43	CLA	Cache locking available 0 Unavailable 1 Available

Table 6-22. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
44	CPA	Cache parity available 0 Unavailable 1 Available
45–49	—	Reserved, should be cleared.
50–52	CNWAY	Cache number of ways 111 Indicates 8 ways
53–55	—	Reserved, should be cleared.
56–63	CSIZE	Cache size 0x20 indicates 32 Kbytes

6.11.4 L1 Cache Configuration Register 1 (L1CFG1)

SPR 516 Access: User read only

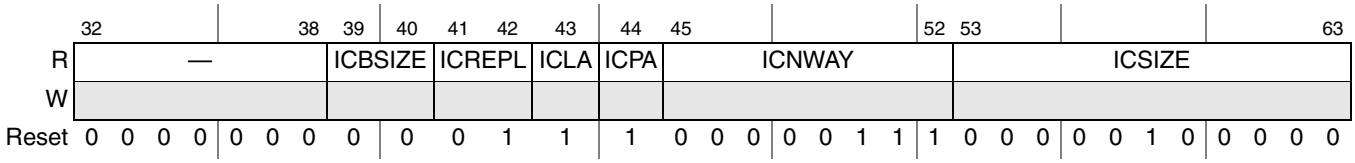


Figure 6-36. L1 Cache Configuration Register 1 (L1CFG1)

Table 6-23. L1CFG1 Field Descriptions

Bits	Name	Description
32–38	—	Reserved, should be cleared.
39–40	ICBSIZ	Instruction cache block size 00 Indicates block size of 32 bytes
41–42	ICREPL	Instruction cache replacement policy 01 Indicates pseudo-LRU policy
43	ICLA	Instruction cache locking available 1 Indicates available
44	ICPA	Instruction cache parity available 1 Indicates available
45–52	ICNWAY	Instruction cache number of ways 111 Indicates 8 ways
53–63	ICSIZE	Instruction cache size 0x20 indicates 32 Kbytes

6.12 MMU Registers

6.12.1 Process ID Registers (PID0–PID2)

SPR 48 (PID0)
SPR 633 (PID1)
SPR 634 (PID2)

Access: Supervisor read/write

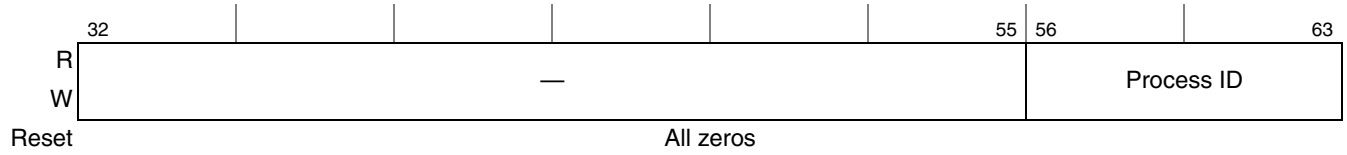


Figure 6-37. Process ID Registers (PID0–PID2)

6.12.2 MMU Control and Status Register 0 (MMUCSR0)

SPR 1012

Access: Supervisor read/write

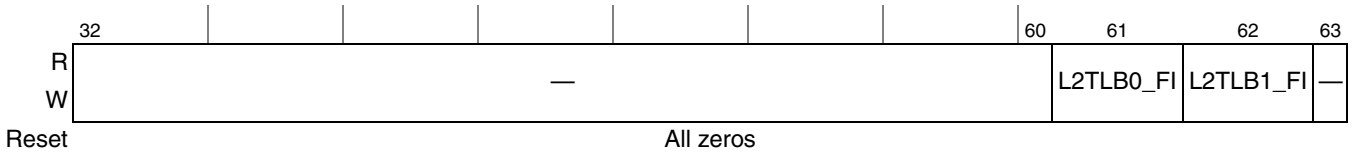


Figure 6-38. MMU Control and Status Register 0 (MMUCSR0)

Table 6-24. MMUCSR0 Field Descriptions

Bits	Name	Description
32–60	—	Reserved, should be cleared.
61	L2TLB0_FI	TLB0 flash invalidate (write to 1 to invalidate)
62	L2TLB1_FI	TLB1 flash invalidate (write 1 to invalidate) 0 No flash invalidate. Writing a 0 to this bit during an invalidation operation is ignored. 1 TLB1 invalidation operation. Hardware initiates a TLB1 invalidation operation. When this operation is complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation.
63	—	Reserved, should be cleared.

6.12.3 MMU Configuration Register (MMUCFG)

SPR 1015

Access: Supervisor read only

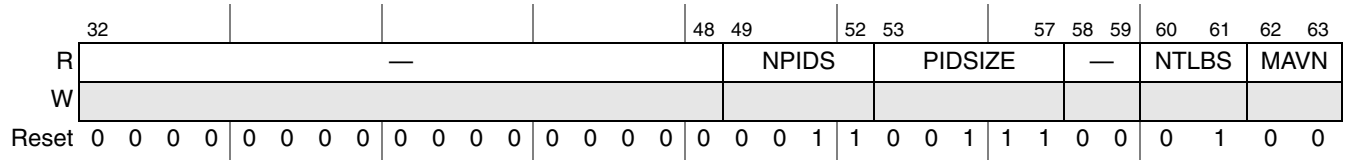


Figure 6-39. MMU Configuration Register (MMUCFG)

Table 6-25. MMUCFG Field Descriptions

Bits	Name	Description
32–48	—	Reserved, should be cleared.
49–52	NPIDS	Number of PID registers, a 4-bit field that indicates the number of PID registers provided by the processor. The e500 implements three PIDs.
53–57	PIDSIZE	PID register size. The 5-bit value of PIDSIZE is one less than the number of bits in each of the PID registers implemented by the processor. The processor implements only the least significant PIDSIZE+1 bits in the PID registers. 00111 indicates 8-bit registers. This is the value presented by the e500.
58–59	—	Reserved, should be cleared.
60–61	NTLBS	Number of TLBs. The value of NTLBS is one less than the number of software-accessible TLB structures that are implemented by the processor. NTLBS is set to one less than the number of TLB structures so that its value matches the maximum value of MAS0[TLBSEL]. 00 1 TLB 01 2 TLBs. This is the value presented by the e500. 10 3 TLBs 11 4 TLBs
62–63	MAVN	MMU architecture version number. Indicates the version number of the architecture of the MMU implemented by the processor. 0b00 indicates version 1.0.

6.12.4 TLB Configuration Registers (TLB*n*CFG)

6.12.4.1 TLB0 Configuration Register 0 (TLB0CFG)

SPR 688 Access: Supervisor read only

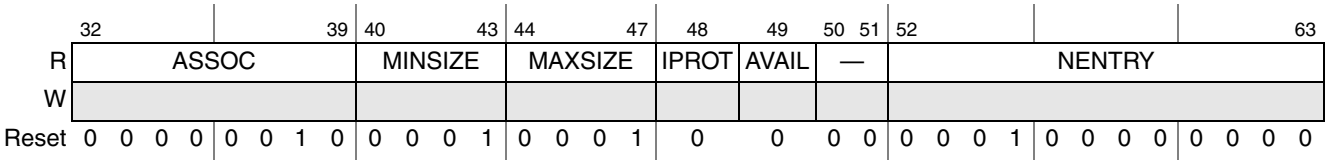


Figure 6-40. TLB Configuration Register 0 (TLB0CFG)

Table 6-26. TLB0CFG Field Descriptions

Bits	Name	Description
32–39	ASSOC	Associativity of TLB0 0x02 indicates associativity is 2-way set associative
40–43	MINSIZE	Minimum page size of TLB0 0x1 indicates smallest page size is 4K
44–47	MAXSIZE	Maximum page size of TLB0 0x1 indicates maximum page size is 4K
48	IPROT	Invalidate protect capability of TLB0 0 Indicates invalidate protection capability not supported
49	AVAIL	Page size availability of TLB0 0 No variable-sized pages available (MINSIZE = MAXSIZE)

Table 6-26. TLB0CFG Field Descriptions (continued)

Bits	Name	Description
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB0 0x100 LB0 contains 256 entries

6.12.4.2 TLB1 Configuration Register 1 (TLB1CFG)

SPR 689 Access: Supervisor read only

	32–39	40–43	44–47	48	49	50 51	52–63
R	ASSOC	MINSIZE	MAXSIZE	IPROT	AVAIL	—	NENTRY
W							
Reset	0 0 0 1	0 0 0 0	0 0 0 1	1 0 0 1	1	1	0 0	0 0 0 0	0 0 0 1	0 0 0 0

Figure 6-41. TLB Configuration Register 1 (TLB1CFG)

Table 6-27. TLB1CFG Field Descriptions

Bits	Name	Description
32–39	ASSOC	Associativity of TLB1 0x10 indicates associativity is 16
40–43	MINSIZE	Minimum page size of TLB1 0x1 indicates smallest page size is 4K
44–47	MAXSIZE	Maximum page size of TLB1 0x9 Indicates maximum page size is 256 Mbyte
48	IPROT	Invalidate protect capability of TLB1 1 Indicates that TLB1 supports invalidate protection capability
49	AVAIL	Page size availability of TLB1 1 Indicates all page sizes between MINSIZE and MAXSIZE supported
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB1 0x010: TLB1 contains 16 entries

6.12.5 MMU Assist Registers

6.12.5.1 MAS Register 0 (MAS0)

SPR 624 Access: Supervisor read/write

	32–34	35	36–43	44–47	48–62	63
R W	—	TLBSEL	—	ESEL	—	NV
Reset	All zeros					

Figure 6-42. MAS Register 0 (MAS0)

Table 6-28. MAS0 Field Descriptions—MMU Read/Write and Replacement Control

Bits	Name	Descriptions
32–34	—	Reserved, should be cleared.
35	TLBSEL	Selects TLB for access 0 TLB0 1 TLB1
36–43	—	Reserved, should be cleared.
44–47	ESEL	Entry select. Number of entry in selected array to be used for **tlbwe**. This field is also updated on TLB error exceptions (misses), and **tlbsx** hit and miss cases. Only certain bits are valid, depending on the array selected in TLBSEL. Other bits should be 0. For the e500, ESEL serves as the way select for the corresponding TLB as follows: When TLBSEL = 00 (TLB0 selected), only bit 47 is used (and bits 44–46 should be cleared). This bit selects between way 0 and way 1 of TLB0. EA bits 45–51 from MAS2[EPN] are used to index into the TLB to further select the entry for the operation. When TLBSEL = 01 (TLB1 selected), all four bits are used to select one of 16 entries in the array.
48–62	—	Reserved, should be cleared.
63	NV	Next victim. Next victim bit value to be written to TLB0[NV] on execution of **tlbwe**. This field is also updated on TLB error exceptions (misses), **tlbsx** hit and miss cases and on execution of **tlbre**. This field is updated based on the calculated next victim bit for TLB0 (based on the round-robin replacement algorithm.) Note that this field is not defined for operations that specify TLB1 (when TLBSEL = 01).

6.12.5.2 MAS Register 1 (MAS1)

SPR 625 Access: Supervisor read/write

	32	33	34–39	40–47	48–50	51	52–55	56–63
R W	V	IPROT	—	TID	—	TS	TSIZE	—
Reset	All zeros							

Figure 6-43. MAS Register 1 (MAS1)

Table 6-29. MAS1 Field Descriptions—Descriptor Context and Configuration Control

Bits	Name	Descriptions
32	V	TLB valid bit 0 This TLB entry is invalid. 1 This TLB entry is valid.
33	IPROT	Invalidate protect. Set to protect this TLB entry from invalidate operations due the execution of **tlbiva**[**x**] (TLB1 only). Note that not all TLB arrays are necessarily protected from invalidation with IPROT. Arrays that support invalidate protection are denoted as such in the TLB configuration registers. 0 Entry is not protected from invalidation 1 Entry is protected from invalidation.
34–39	—	Reserved, should be cleared.
40–47	TID	Translation identity. An 8-bit field that defines the process ID for this TLB entry. TID is compared with the current process IDs of the three virtual address to be translated. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 6-29. MAS1 Field Descriptions—Descriptor Context and Configuration Control (continued)

Bits	Name	Descriptions
48–50	—	Reserved, should be cleared.
51	TS	Translation space. This bit is compared with the IS or DS fields of the MSR (depending on the type of access) to determine if this TLB entry may be used for translation.
52–55	TSIZE	Translation size. Defines the TLB entry page size. For arrays that contain fixed-size TLB entries, TSIZE is ignored. For variable page size arrays, the page size is 4^{TSIZE} Kbytes. The e500 supports the following sizes: 0001 4 Kbytes 0010 16 Kbytes 0011 64 Kbytes 0100 256 Kbytes 0101 1 Mbyte 0110 4 Mbytes 0111 16 Mbytes 1000 64 Mbytes 1001 256 Mbytes
56–63	—	Reserved, should be cleared.

6.12.5.3 MAS Register 2 (MAS2)

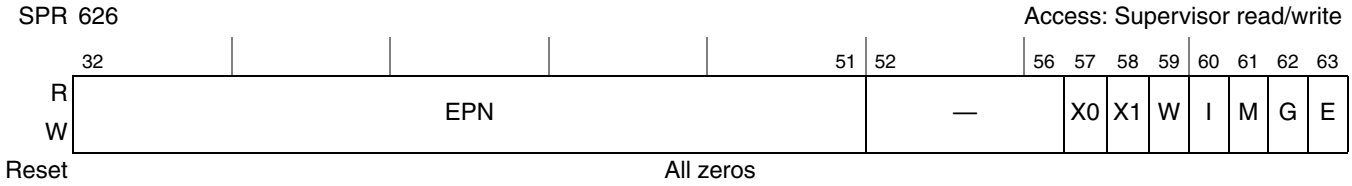


Figure 6-44. MAS Register 2 (MAS2)

Table 6-30. MAS2 Field Descriptions—EPN and Page Attributes

Bits	Name	Description
32–51	EPN	Effective page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared.
52–56	—	Reserved for implementation-specific use
57	X0	Implementation-dependent page attribute
58	X1	Implementation-dependent page attribute
59	W	Write-through 0 This page is considered write-back with respect to the caches in the system. 1 All stores performed to this page are written through the caches to main memory.
60	I	Caching-inhibited 0 Accesses to this page are considered cacheable. 1 The page is considered caching-inhibited. All loads and stores to the page bypass the caches and are performed directly to main memory.
61	M	Memory coherency required 0 Memory coherency is not required. 1 Memory coherency is required. This allows loads and stores to this page to be coherent with loads and stores from other processors (and devices) in the system, assuming all such devices are participating in the coherency protocol.

Table 6-30. MAS2 Field Descriptions—EPN and Page Attributes (continued)

Bits	Name	Description
62	G	Guarded 0 Accesses to this page are not guarded and can be performed before it is known if they are required by the sequential execution model. 1 All loads and stores to this page that miss in the L1 cache are performed without speculation (that is, they are known to be required). Speculative loads can be performed if they hit in the L1 cache. In addition, accesses to caching-inhibited pages are performed using only the memory element that is explicitly specified.
63	E	Endianness. Determines endianness for the corresponding page. Little-endian operation is true little endian, which differs from the modified little-endian byte-ordering model optionally available in previous devices that implement the original PowerPC architecture. See the *PowerPC™ e500 Core Family Reference Manual* for more information. 0 The page is accessed in big-endian byte order. 1 The page is accessed in true little-endian byte order.

6.12.5.4 MAS Register 3 (MAS3)

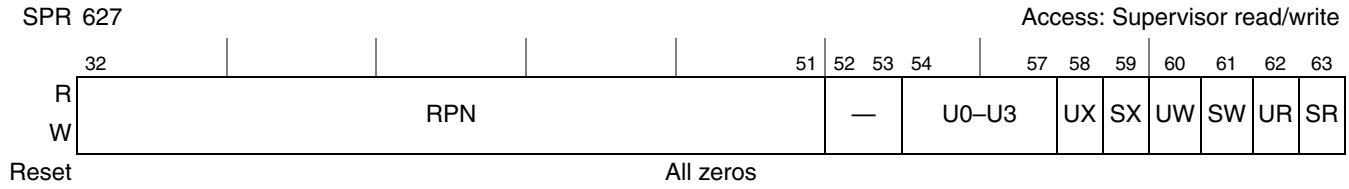


Figure 6-45. MAS Register 3 (MAS3)

Table 6-31. MAS3 Field Descriptions–RPN and Access Control

Bits	Name	Description
32–51	RPN	Real page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared.
52–53	—	Reserved, should be cleared.
54–57	U0–U3	User attribute bits. Associated with a TLB entry and can be used by system software. For example, they can hold information useful to a page-scanning algorithm or mark more abstract page attributes.
58–63	PERMIS	Permission bits (UX, SX, UW, SW, UR, SR). User and supervisor read, write, and execute permission bits.

6.12.5.5 MAS Register 4 (MAS4)

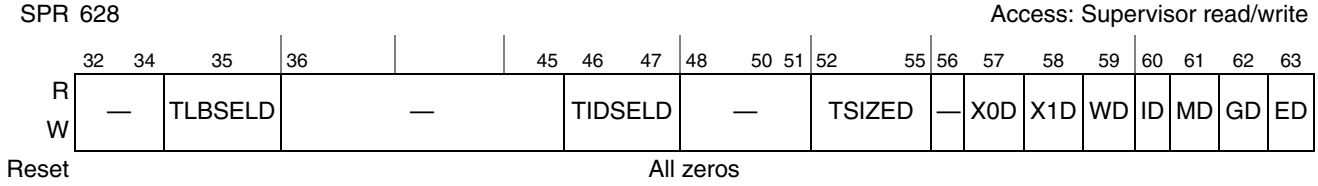


Figure 6-46. MAS Register 4 (MAS4)

Table 6-32. MAS4 Field Descriptions—Hardware Replacement Assist Configuration

Bits	Name	Description
32–34	—	Reserved, should be cleared.
35	TLBSELD	TLBSEL default value. The default value to be loaded in MAS0[TLBSEL] on a TLB miss exception. 0 TLB0 1 TLB1
36–45	—	Reserved, should be cleared.
46–47	TIDSELD	TID default selection value. A 2-bit field that specifies which of the current PID registers should be used to load the MAS1[TID] field on a TLB miss exception.
		The e500 implementation defines this field as follows: 00 PID0 01 PID1 10 PID2 11 TIDZ (0x00) (all zeros)
48–51	—	Reserved, should be cleared.
52–55	TSIZED	Default TSIZE value. Specifies the default value to be loaded into MAS1[TSIZE] on a TLB miss exception.
56	—	Reserved, should be cleared.
57	X0D	Default X0 value. Specifies the default value to be loaded into MAS2[X0] on a TLB miss exception.
58	X1D	Default X1 value. Specifies the default value to be loaded into MAS2[X1] on a TLB miss exception.
59	WD	Default W value. Specifies the default value to be loaded into MAS2[W] on a TLB miss exception.
60	ID	Default I value. Specifies the default value to be loaded into MAS2[I] on a TLB miss exception.
61	MD	Default M value. Specifies the default value to be loaded into MAS2[M] on a TLB miss exception.
62	GD	Default G value. Specifies the default value to be loaded into MAS2[G] on a TLB miss exception.
63	ED	Default E value. Specifies the default value to be loaded into MAS2[E] on a TLB miss exception.

6.12.5.6 MAS Register 6 (MAS6)

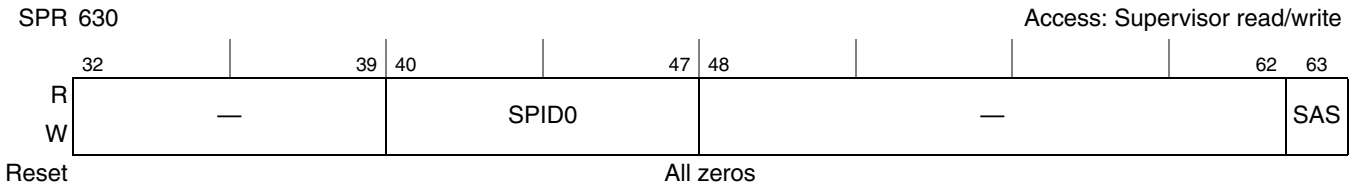


Figure 6-47. MAS Register 6 (MAS6)

Table 6-33. MAS6—TLB Search Context Register 0

Bits	Name	Comments, or Function when Set
32–39	—	Reserved, should be cleared.
40–47	SPID0	Specifies the PID value (recent value of PID0) used when searching the TLB during execution of **tlbsx**.
48–62	—	Reserved, should be cleared.
63	SAS	Address space (AS) value for searches. Specifies the value of AS used when searching the TLB (during execution of **tlbsx**).

6.13 Debug Registers

6.13.1 Debug Control Registers (DBCR0–DBCR2)

6.13.1.1 Debug Control Register 0 (DBCR0)

SPR 308 Access: Supervisor read/write

	32	33	34 35	36	37	38	39	40	41	42 43	44 45	46 47	48	49 — 62	63
R W	—	IDM	RST	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	—	DAC1	DAC2	RET	—	FT
Reset	All zeros														

Figure 6-48. Debug Control Register 0 (DBCR0)

Table 6-34. DBCR0 Field Descriptions

Bits	Name	Description
32	—	Reserved, should be cleared.
33	IDM	Internal debug mode 0 Debug interrupts are disabled. No debug interrupts are taken and debug events are not logged. 1 If MSR[DE] = 1, the occurrence of a debug event or the recording of an earlier debug event in the DBSR when MSR[DE] = 0 or DBCR0[IDM] = 0 causes a debug interrupt. **Programming note:** Software must clear debug event status in the DBSR in the debug interrupt handler when a debug interrupt is taken before re-enabling interrupts through MSR[DE]. Otherwise, redundant debug interrupts are taken for the same debug event.
34–35	RST	Reset. The e500 implements these bits as follows: 0*x* Default (No action) 1*x* Causes a hard reset if MSR[DE] and DBCR0[IDM] are set. Always cleared on subsequent cycle. This causes a hard reset to the core only.
36	ICMP	Instruction completion debug event enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled Note: Instruction completion does not cause an ICMP debug event if MSR[DE] = 0.
37	BRT	Branch taken debug event enable 0 BRT debug events are disabled 1 BRT debug events are enabled Note: Taken branches do not cause a BRT debug event if MSR[DE] = 0.
38	IRPT	Interrupt taken debug event enable. This bit affects only noncritical interrupts. 0 IRPT debug events are disabled 1 IRPT debug events are enabled
39	TRAP	Trap debug event enable 0 TRAP debug events cannot occur 1 TRAP debug events can occur
40	IAC1	Instruction address compare 1 debug event enable 0 IAC1 debug events cannot occur 1 IAC1 debug events can occur

Table 6-34. DBCR0 Field Descriptions (continued)

Bits	Name	Description
41	IAC2	Instruction address compare 2 debug event enable 0 IAC2 debug events cannot occur 1 IAC2 debug events can occur
42–43	—	Reserved, should be cleared.
44–45	DAC1	Data address compare 1 debug event enable 00 DAC1 debug events cannot occur 01 DAC1 debug events can occur only if a store-type data storage access 10 DAC1 debug events can occur only if a load-type data storage access 11 DAC1 debug events can occur on any data storage access
46–47	DAC2	Data address compare 2 debug event enable 00 DAC2 debug events cannot occur 01 DAC2 debug events can occur only if a store-type data storage access 10 DAC2 debug events can occur only if a load-type data storage access 11 DAC2 debug events can occur on any data storage access
48	RET	Return debug event enable 0 RET debug events cannot occur 1 RET debug events can occur Note: An **rfci** does not cause an RET debug event if MSR[DE] = 0 at the time that **rfci** executes.
49–62	—	Reserved, should be cleared.
63	FT	Freeze timers on debug event 0 Enable clocking of timers 1 Disable clocking of timers if any DBSR bit is set (except MRR)

6.13.1.2 Debug Control Register 1 (DBCR1)

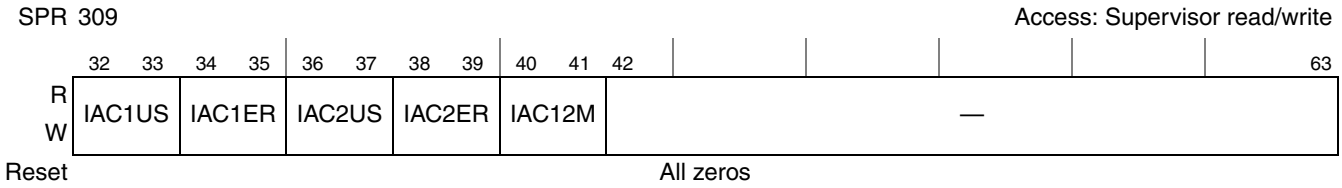


Figure 6-49. Debug Control Register 1 (DBCR1)

Table 6-35. DBCR1 Field Descriptions

Bits	Name	Description
32–33	IAC1US	Instruction address compare 1 user/supervisor mode 00 IAC1 debug events can occur 01 Reserved 10 IAC1 debug events can occur only if MSR[PR] = 0 11 IAC1 debug events can occur only if MSR[PR] = 1
34–35	IAC1ER	Instruction address compare 1 effective/real mode 00 IAC1 debug events are based on effective addresses 01 Reserved on the e500 10 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 1

Table 6-35. DBCR1 Field Descriptions (continued)

Bits	Name	Description
36–37	IAC2US	Instruction address compare 2 user/supervisor mode 00 IAC2 debug events can occur 01 Reserved 10 IAC2 debug events can occur only if MSR[PR] = 0 11 IAC2 debug events can occur only if MSR[PR] = 1
38–39	IAC2ER	Instruction address compare 2 effective/real mode 00 IAC2 debug events are based on effective addresses 01 Reserved on the e500 10 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 1
40–41	IAC12M	Instruction address compare 1/2 mode 00 Exact address compare. IAC1 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 and IAC2 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, plus ANDed with the contents of IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined. 10 Inclusive address range compare. IAC1 and IAC2 debug events occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined. 11 Exclusive address range compare. IAC1 and IAC2 debug events occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined.
42–63	—	Reserved, should be cleared.

6.13.1.3 Debug Control Register 2 (DBCR2)

SPR 310 Access: Supervisor read/write

	32–33	34–35	36–37	38–39	40–41	42–63
R W	DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	—
Reset	All zeros					

Figure 6-50. Debug Control Register 2 (DBCR2)

Table 6-36. DBCR2 Field Descriptions

Bits	Name	Description
32–33	DAC1US	Data address compare 1 user/supervisor mode 00 DAC1 debug events can occur 01 Reserved 10 DAC1 debug events can occur only if MSR[PR] = 0. 11 DAC1 debug events can occur only if MSR[PR] = 1.

Table 6-36. DBCR2 Field Descriptions (continued)

Bits	Name	Description
34–35	DAC1ER	Data address compare 1 effective/real mode 00 DAC1 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
36–37	DAC2US	Data address compare 2 user/supervisor mode 00 DAC2 debug events can occur. 01 Reserved 10 DAC2 debug events can occur only if MSR[PR] = 0. 11 DAC2 debug events can occur only if MSR[PR] = 1.
38–39	DAC2ER	Data address compare 2 effective/real mode 00 DAC2 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
40–41	DAC12M	Data address compare 1/2 mode 00 Exact address compare. DAC1 debug events can occur only if the address of the data storage access is equal to the value specified in DAC1. DAC2 debug events can occur only if the address of the data storage access is equal to the value specified in DAC2. 01 Address bit match. DAC1 and DAC2 debug events can occur only if the address of the data storage access, ANDed with the contents of DAC2 are equal to the contents of DAC1, also ANDed with the contents of DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 10 Inclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 11 Exclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined.
42–63	—	Reserved, should be cleared.

6.13.2 Debug Status Register (DBSR)

SPR: 304 Access: Supervisor w1c

	32	33	34 35	36	37	38	39	40	41	42 43	44	45	46	47
R	IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	—	DAC1R	DAC1W	DAC2R	DAC2W
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c
Reset	0	0	undefined	0	0	0	0	0	0	0 0	0	0	0	0

	48	49	63
R	RET	—	
W	w1c		
Reset	All zeros		

Figure 6-51. Debug Status Register (DBSR)

Table 6-37. DBSR Field Descriptions

Bits	Name	Description
32	IDE	Imprecise debug event. Set if MSR[DE] = 0 and a debug event causes its respective DBSR bit to be set. Functions as write-one-to-clear.
33	UDE	Unconditional debug event. Set if an unconditional debug event occurred. Functions as write-one-to-clear. If $\overline{\text{UDE}}$ (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: MSR[DE]DBCR0[IDM]Action X 0 No action. 0 1 UDE is set. 1 1 UDE is set and a debug interrupt is taken.
34–35	MRR	Most recent reset. Functions as write-one-to-clear. Undefined at power-on. The e500 implements HRESET as follows: 0x No hard reset occurred since this bit was last cleared by software. 1x The previous reset was a hard reset.
36	ICMP	Instruction complete debug event. Set if an instruction completion debug event occurred and DBCR0[ICMP] = 1. Functions as write-one-to-clear.
37	BRT	Branch taken debug event. Set if a branch taken debug event occurred (DBCR0[BRT] = 1). Functions as write-one-to-clear.
38	IRPT	Interrupt taken debug event. Set if an interrupt taken debug event occurred (DBCR0[IRPT] = 1). Functions as write-one-to-clear.
39	TRAP	Trap instruction debug event. Set if a trap Instruction debug event occurred (DBCR0[TRAP] = 1). Functions as write-one-to-clear.
40	IAC1	Instruction address compare 1 debug event. Set if an IAC1 debug event occurred (DBCR0[IAC1] = 1). Functions as write-one-to-clear.
41	IAC2	Instruction address compare 2 debug event. Set if an IAC2 debug event occurred (DBCR0[IAC2] = 1). Functions as write-one-to-clear.
42–43	—	Reserved, should be cleared
44	DAC1R	Data address compare 1 read debug event. Set if a read-type DAC1 debug event occurred (DBCR0[DAC1] = 10 or 11). Functions as write-one-to-clear.
45	DAC1W	Data address compare 1 write debug event. Set if a write-type DAC1 debug event occurred (DBCR0[DAC1] = 01 or 11). Functions as write-one-to-clear.
46	DAC2R	Data address compare 2 read debug event.Set if a read-type DAC2 debug event occurred (DBCR0[DAC2] = 10 or 11). Functions as write-one-to-clear.
47	DAC2W	Data address compare 2 write debug event. Set if a write-type DAC2 debug event occurred (DBCR0[DAC2] = 01 or 11). Functions as write-one-to-clear.
48	RET	Return debug event. Set if a return debug event occurred (DBCR0[RET] = 1). Functions as write-one-to-clear.
49–63	—	Reserved, should be cleared.

6.13.3 Instruction Address Compare Registers (IAC1–IAC2)

SPR 312 (IAC1); SPR 313 (IAC2) Access: Supervisor read/write

	32							61	62	63
R W	Instruction address								—	
Reset	All zeros									

Figure 6-52. Instruction Address Compare Registers (IAC1–IAC2)

6.13.4 Data Address Compare Registers (DAC1–DAC2)

SPR 316 (DAC1); SPR 317 (DAC2) Access: Supervisor read/write

	32							63
R W	Data address							
Reset	All zeros							

Figure 6-53. Data Address Compare Registers (DAC1–DAC2)

6.14 Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

SPR: 512 Access: User mixed

	High-Word Error Bits									Status Bits						
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
R W	SOVH	OVH	FGH	FXH	FINVH	FDBZH	FUNFH	FOVFH	—		FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE
Reset	All zeros															

	Enable Bits															
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R W	SOV	OV	FG	FX	FINV	FDBZ	FUNF	FOVF	—	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	
Reset	All zeros															

Figure 6-54. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

Table 6-38. SPEFSCR Field Descriptions

Bits	Name	Function
32	SOVH	Summary integer overflow high. Set whenever an instruction (except **mtspr**) sets OVH. SOVH remains set until it is cleared by an **mtspr[SPEFSCR]**.
33	OVH	Integer overflow high. An overflow occurred in the upper half of the register while executing a SPE integer instruction

Table 6-38. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
34	FGH	Embedded floating-point guard bit high. Floating-point guard bit from the upper half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
35	FXH	Embedded floating-point sticky bit high. Floating bit from the upper half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
36	FINVH	Embedded floating-point invalid operation error high. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
37	FDBZH	Embedded floating-point divide by zero error high. Set if the dividend is non-zero and the divisor is zero.
38	FUNFH	Embedded floating-point underflow error high
39	FOVFH	Embedded floating-point overflow error high
40–41	—	Reserved, should be cleared.
42	FINXS	Embedded floating-point inexact sticky. FINXS = FINXS | FGH | FXH | FG | FX
43	FINVS	Embedded floating-point invalid operation sticky. Location for software to use when implementing true IEEE floating point.
44	FDBZS	Embedded floating-point divide by zero sticky. FDBZS = FDBZS | FDBZH | FDBZ
45	FUNFS	Embedded floating-point underflow sticky. Storage location for software to use when implementing true IEEE floating point.
46	FOVFS	Embedded floating-point overflow sticky. Storage location for software to use when implementing true IEEE floating point.
47	MODE	Embedded floating-point mode (read only on e500)
48	SOV	Integer summary overflow. Set whenever an SPE instruction (except **mtspr**) sets OV. SOV remains set until it is cleared by **mtspr[SPEFSCR]**.
49	OV	Integer overflow. An overflow occurred in the lower half of the register while a SPE integer instruction is being executed.
50	FG	Embedded floating-point guard bit. Floating-point guard bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
51	FX	Embedded floating-point sticky bit. Floating bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
52	FINV	Embedded floating-point invalid operation error. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
53	FDBZ	Embedded floating-point divide by zero error. Set of the dividend is non-zero and the divisor is zero.
54	FUNF	Embedded floating-point underflow error
55	FOVF	Embedded floating-point overflow error
56	—	Reserved, should be cleared.
57	FINXE	Embedded floating-point inexact enable

Table 6-38. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
58	FINVE	Embedded floating-point invalid operation/input error exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FINV or FINVH is set by a floating-point instruction.
59	FDBZE	Embedded floating-point divide-by-zero exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FDBZ or FDBZH is set by a floating-point instruction.
60	FUNFE	Embedded floating-point underflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FUNF or FUNFH is set by a floating-point instruction.
61	FOVFE	Embedded floating-point overflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FOVF or FOVFH is set by a floating-point instruction.
62–63	FRMC	Embedded floating-point rounding mode control 00 Round to nearest 01 Round toward zero 10 Round toward +infinity 11 Round toward –infinity

6.14.1 Accumulator (ACC)

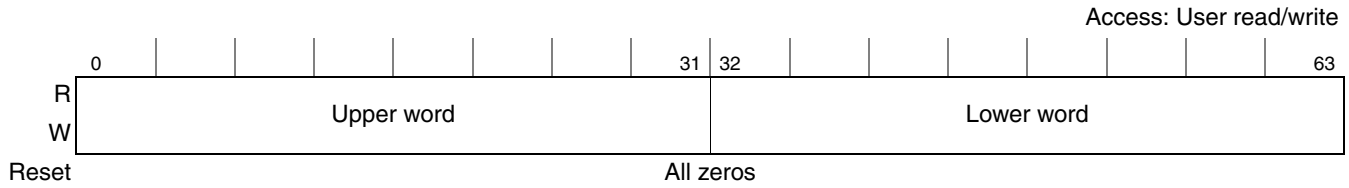


Figure 6-55. Accumulator (ACC)

Table 6-39. ACC Fleld Descriptions

Bits	Name	Function
0–31	Upper word	Holds the upper-word accumulate value for SPE multiply with accumulate instructions
32–63	Lower word	Holds the lower-word accumulate value for SPE multiply with accumulate instructions

6.15 Performance Monitor Registers (PMRs)

Table 6-40. Supervisor-Level PMRs (PMR[5] = 1)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
PMC0	Performance monitor counter 0	16	00000	10000	6.15.4/6-50
PMC1	Performance monitor counter 1	17	00000	10001	
PMC2	Performance monitor counter 2	18	00000	10010	
PMC3	Performance monitor counter 3	19	00000	10011	
PMGC0	Performance monitor global control register 0	400	01100	10000	6.15.1/6-48
PMLCa0	Performance monitor local control a0	144	00100	10000	6.15.2/6-48
PMLCa1	Performance monitor local control a1	145	00100	10001	
PMLCa2	Performance monitor local control a2	146	00100	10010	
PMLCa3	Performance monitor local control a3	147	00100	10011	
PMLCb0	Performance monitor local control b0	272	01000	10000	6.15.3/6-49
PMLCb1	Performance monitor local control b1	273	01000	10001	
PMLCb2	Performance monitor local control b2	274	01000	10010	
PMLCb3	Performance monitor local control b3	275	01000	10011	

Table 6-41. User-Level PMRs (PMR[5] = 0) (Read Only)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
UPMC0	User performance monitor counter 0	0	00000	00000	6.15.4/6-50
UPMC1	User performance monitor counter 1	1	00000	00001	
UPMC2	User performance monitor counter 2	2	00000	00010	
UPMC3	User performance monitor counter 3	3	00000	00011	
UPMLCa0	User performance monitor local control a0	128	00100	00000	6.15.3/6-49
UPMLCa1	User performance monitor local control a1	129	00100	00001	
UPMLCa2	User performance monitor local control a2	130	00100	00010	
UPMLCa3	User performance monitor local control a3	131	00100	00011	
UPMLCb0	User performance monitor local control b0	256	01000	00000	6.15.3/6-49
UPMLCb1	User performance monitor local control b1	257	01000	00001	
UPMLCb2	User performance monitor local control b2	258	01000	00010	
UPMLCb3	User performance monitor local control b3	259	01000	00011	
UPMGC0	User performance monitor global control register 0	384	01100	00000	6.15.2/6-48

6.15.1 Global Control Register 0 (PMGC0, UPMGC0)

PMGC0 (PMR400)
UPMGC0 (PMR384)

Access: PMGC0: Supervisor- read/write
UPMGC0: Supervisor/user read only

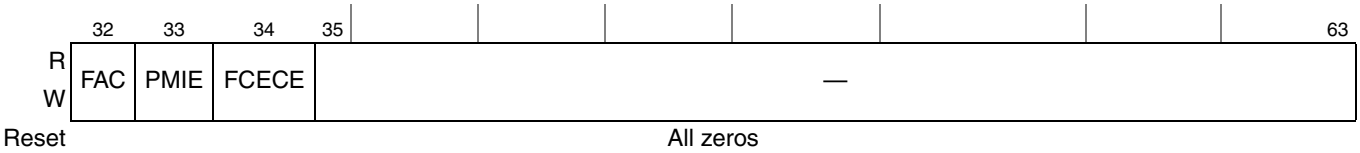


Figure 6-56. Performance Monitor Global Control Register 0 (PMGC0), User Performance Monitor Global Control Register 0 (UPMGC0)

Table 6-42. PMGC0 Field Descriptions

Bits	Name	Description
32	FAC	Freeze all counters. When FAC is set by hardware or software, PMLCx[FC] maintains its current value until it is changed by software. 0 The PMCs are incremented (if permitted by other PM control bits). 1 The PMCs are not incremented.
33	PMIE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled and occur when an enabled condition or event occurs.
34	FCECE	Freeze counters on enabled condition or event 0 The PMCs can be incremented (if permitted by other PM control bits). 1 The PMCs can be incremented (if permitted by other PM control bits) only until an enabled condition or event occurs. When an enabled condition or event occurs, PMGC0[FAC] is set. It is up to software to clear FAC.
35–63	—	Reserved, should be cleared.

6.15.2 Local Control A Registers (PMLCa0–PMLCa3, UPMLCa0–UPMLCa3)

PMLCa0 (PMR144)
PMLCa1 (PMR145)
PMLCa2 (PMR146)
PMLCa3 (PMR147)

UPMLCa0 (PMR128)
UPMLCa1 (PMR129)
UPMLCa2 (PMR130)
UPMLCa3 (PMR131)

Access: PMLCa0–PMLCa3: Supervisor read/write
UPMLCa0–UPMLCa3: Supervisor/user read only

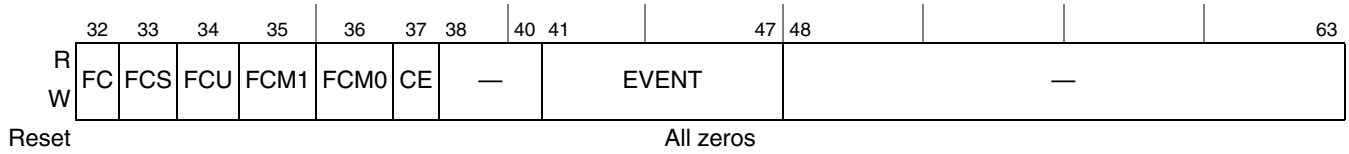


Figure 6-57. Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3)

Table 6-43. PMLCa0–PMLCa3 Field Descriptions

Bits	Name	Description
32	FC	Freeze counter 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented.
33	FCS	Freeze counter in supervisor state 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PR] = 0.
34	FCU	Freeze counter in user state 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PR] = 1.
35	FCM1	Freeze counter while mark = 1 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PMM] = 1.
36	FCM0	Freeze counter while mark = 0 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PMM] = 0.
37	CE	Condition enable 0 PMC*x* overflow conditions cannot occur (PMC*x* cannot cause interrupts, cannot freeze counters) 1 Overflow conditions occur when the most-significant bit of PMC*x* is equal to 1. It is recommended that CE be cleared when counter PMC*x* is selected for chaining.
38–40	—	Reserved, should be cleared.
41–47	EVENT	Event selector. Up to 128 events selectable. These events are described in the *PowerPC™ e500 Core Family Reference Manual.*
48–63	—	Reserved, should be cleared.

6.15.3 Local Control B Registers (PMLCb0–PMLCb3, UPMLCb0–UPMLCb3)

PMLCb0 (PMR272) UPMLCb0 (PMR256)
PMLCb1 (PMR273) UPMLCb1 (PMR257)
PMLCb2 (PMR274) UPMLCb2 (PMR258)
PMLCb3 (PMR275) UPMLCb3 (PMR259)

Access: PMLCb0–PMLCb3: Supervisor read/write
UPMLCb0–UPMLCb3: Supervisor/user read only

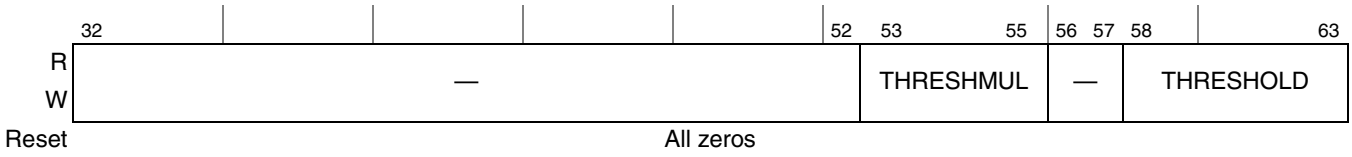


Figure 6-58. Local Control B Registers (PMLCb0–PMLCb3)/User Local Control B Registers (UPMLCb0–UPMLCb3)

Table 6-44. PMLCb0–PMLCb3 Field Descriptions

Bits	Name	Description
32–52	—	Reserved, should be cleared.
53–55	THRESHMUL	Threshold multiple 000 Threshold field is multiplied by 1 (PMLCb*n*[THRESHOLD] × 1) 001 Threshold field is multiplied by 2 (PMLCb*n*[THRESHOLD] × 2) 010 Threshold field is multiplied by 4 (PMLCb*n*[THRESHOLD] × 4) 011 Threshold field is multiplied by 8 (PMLCb*n*[THRESHOLD] × 8) 100 Threshold field is multiplied by 16 (PMLCb*n*[THRESHOLD] × 16) 101 Threshold field is multiplied by 32 (PMLCb*n*[THRESHOLD] × 32) 110 Threshold field is multiplied by 64 (PMLCb*n*[THRESHOLD] × 64) 111 Threshold field is multiplied by 128 (PMLCb*n*[THRESHOLD] × 128)
56–57	—	Reserved, should be cleared.
58–63	THRESHOLD	Threshold. Only events that exceed the threshold value are counted. Such events are implementation-dependent as are the dimension (for example duration in cycles) and granularity with which the value is interpreted. By varying the value, software can obtain a profile of the event characteristics subject to thresholding. For example, if PMC1 is configured to count cache misses that last longer than the threshold value, software can obtain the distribution of cache miss durations for a given program by monitoring the program repeatedly using a different threshold each time.

6.15.4 Performance Monitor Counter Registers (PMC0–PMC3, UPMC0–UPMC3)

PMC0 (PMR16) UPMC0 (PMR0)
PMC1 (PMR17) UPMC1 (PMR1)
PMC2 (PMR18) UPMC2 (PMR2)
PMC3 (PMR19) UPMC3 (PMR3)

Access: PMC0–PMC3: Supervisor read/write
UPMC0–UPMC3: Supervisor/user read only

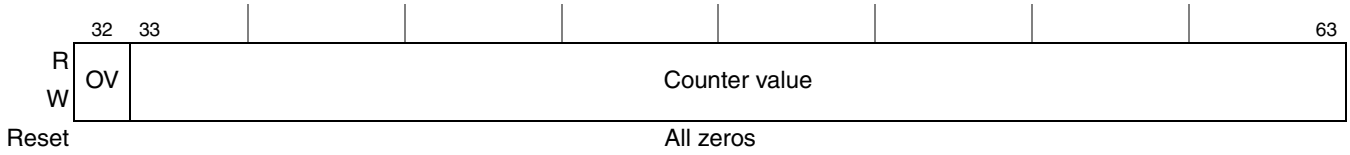


Figure 6-59. Performance Monitor Counter Registers (PMC0–PMC3)/User Performance Monitor Counter Registers (UPMC0–UPMC3)

Table 6-45. PMC0–PMC3 Field Descriptions

Bits	Name	Description
32	OV	Overflow. When this bit is set, it indicates this counter reaches its maximum value.
33–63	Counter value	Indicates the number of occurrences of the specified event

Chapter 7
L2 Look-Aside Cache/SRAM

This chapter describes the organization of the on-chip L2/SRAM, cache coherency rules, cache line replacement algorithm, cache control instructions, and various cache operations. It also describes the interaction between the L2/SRAM and the e500 core complex.

7.1 L2 Cache Overview

The integrated 256-Kbyte L2 cache is organized as 1024 8 way sets of 32-byte cache lines based on 32-bit physical addresses, as shown in Figure 7-1.

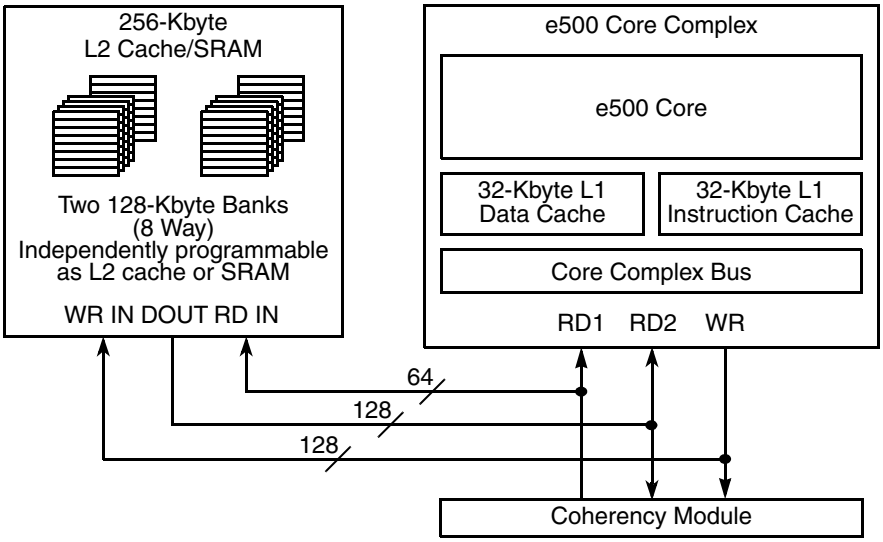


Figure 7-1. L2 Cache/SRAM Configuration

The SRAM can be configured with memory-mapped registers as externally accessible memory-mapped SRAM in addition to or instead of cache. The L2 cache can operate in the following modes, described in Section 7.2, “Cache Organization”:

- Full cache mode (256-Kbyte cache)
- Full memory-mapped SRAM mode (256-Kbyte SRAM mapped as a single 256-Kbyte block or two 128-Kbyte blocks)
- Half SRAM and half cache mode (128-Kbyte cache and 128-Kbyte memory-mapped SRAM)

7.1.1 L2 Cache and SRAM Features

Two 128-Kbyte arrays can be designated independently as externally-accessible memory-mapped SRAM or cache. The L2 cache has the following characteristics:

- Write-through, front-side cache
  - — Front-side design provides easier cache access for I/O masters such as Ethernet and CPM
  - — Write-through design is more efficient on the processor bus for front-side caches
- Valid, locked, and stale states (no modified state)
- Two input data buses (64 and 128 bits) and one output data bus (128 bits wide)
- All accesses are fully pipelined and non-blocking (allows hits under misses)
- 256-Kbyte array organized as 1024 8 way sets of 32-byte cache lines
- Eight-way set-associativity (high level of associativity yields good performance even with many locked lines)
- Tag arrays contain 17 tag bits and 1 tag parity bit per line to support 256-Kbyte cache (1024 sets), or 18 tag bits to support 128-Kbyte cache (512 sets).
- Configurable to allocate processor instructions, data, or both
  - — Allows external writes (stashing) to allocate and optionally lock a line using one of the two following methods:
    - – Attributes attached to the transactions by initiator or ATMU
    - – I/O devices can force memory writes to be allocated using programmed memory ranges
- Pseudo-LRU (7-bit replacement algorithm)
- Data ECC on 64-bit boundaries (single-error correction, double-error detection)
- Tag parity for 256-Kbyte mode (1 tag bit per line covering cache tags)
- Cache locking methods
  - — Individual line locks are set and cleared by using e500 cache locking instructions—Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**).
  - — A lock attribute can be attached to write operations.
  - — Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWAR*n*).
  - — The entire cache can be locked by setting a configuration register appropriately.
- Lock clearing methods
  - — Individual locks cleared by cache locking instructions (Instruction Cache Block Lock Clear (**icblc**) and Data Cache Block Lock Clear (**dcblc**)) or by snooped flush unless entire cache is locked.
  - — Flash clearing of all instruction and/or data locks is done by writes to configuration registers.
  - — An unlock attribute attached to a read instruction.
- Error injection modes for testing

SRAM features include the following:

- I/O devices access SRAM regions by marking transactions as snoopable (global)
- Regions can reside at any aligned location in the memory map
- For accesses of less than a cache-line, byte-accessible ECC is protected using read-modify-write transactions.

7.2 Cache Organization

When the entire 256-Kbyte array is used as a cache, it has two banks each containing 512 sets of eight cache blocks (8 ways), as shown in Figure 7-2. Each block consists of 32 bytes of data and an address tag.

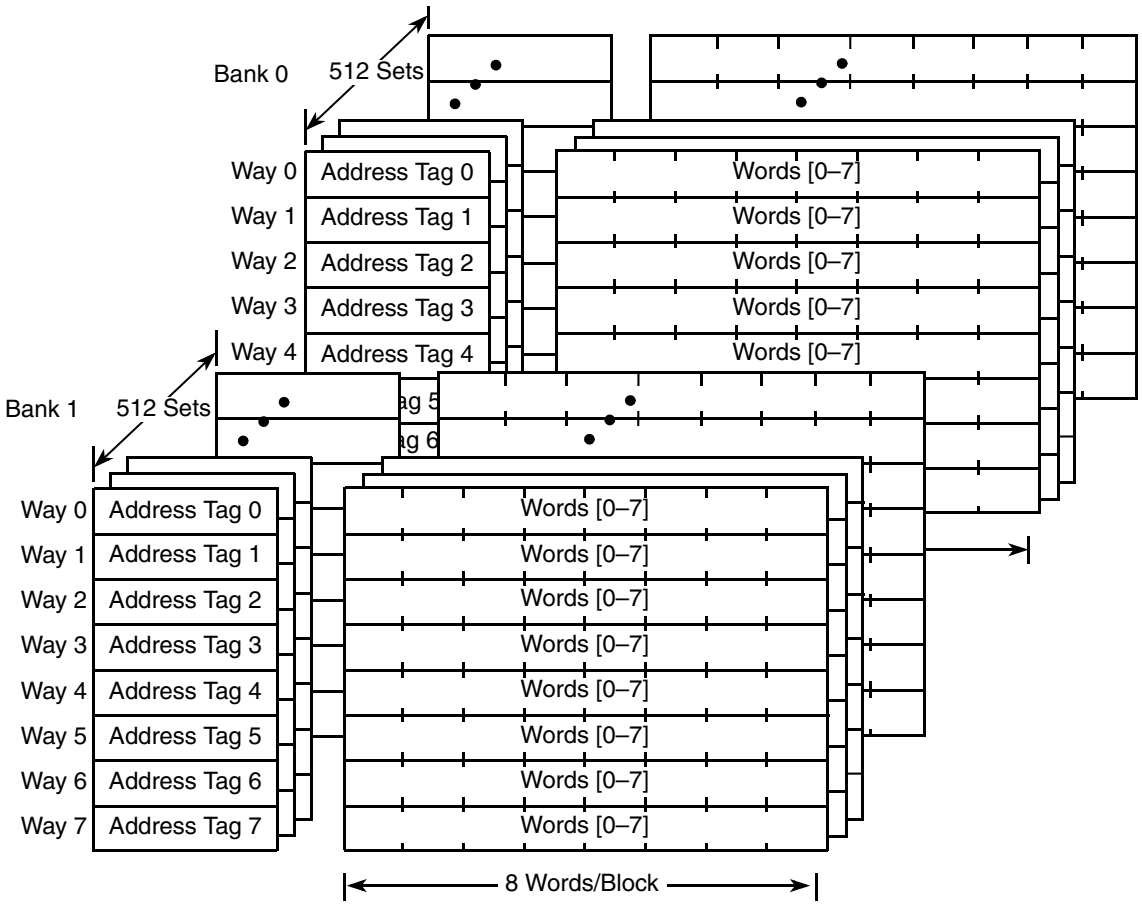


Figure 7-2. Cache Organization

The tag size depends on whether both 128-Kbyte arrays are configured as cache. Figure 7-3 shows how physical address bits are used to access the L2 in full cache mode (256-Kbyte cache).

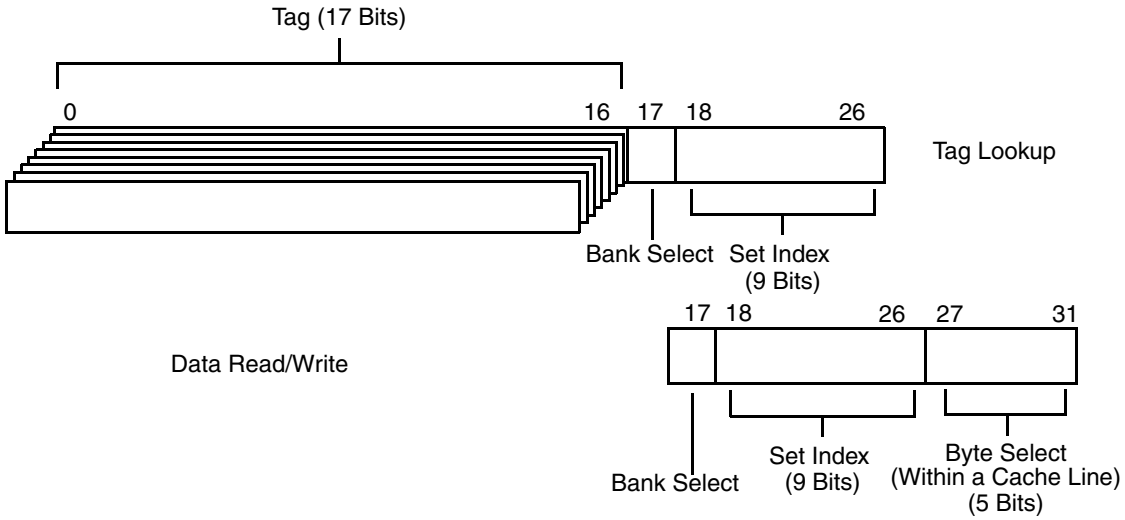


Figure 7-3. 256-Kbyte L2 Cache Address Configuration—Full Cache Mode

Physical address bits 17–26 identify the bank and set of the tag and data. Physical address bits 0–16 are compared against the tags of all 8 ways. A match of a valid tag selects a 32-byte block of data within the set. Physical address bits 27–31 identify the byte or bytes of data within the block. If the L2 is programmed as one block of 256-Kbyte SRAM, physical address bits 14–16 are used as the way select, without a tag lookup.

In half SRAM, half cache mode (128-Kbyte cache), there is no bank select, as shown in Figure 7-4, so the tag is 18 bits (with no parity bit) for cache accesses. The cache resides in bank 1, and the SRAM in bank 0. For SRAM accesses, physical address bits 15–17 are used as the way select. If the L2 is programmed as two blocks of 128-Kbyte SRAM, bank 0 is block 0 (defined by L2SRBAR0, see Table 7-5) and bank 1 is block 1.

As shown in Figure 7-4, the tag is 18 bits (with no parity bit) for cache accesses. For SRAM accesses, physical address bits 15–17 are used as the way select.

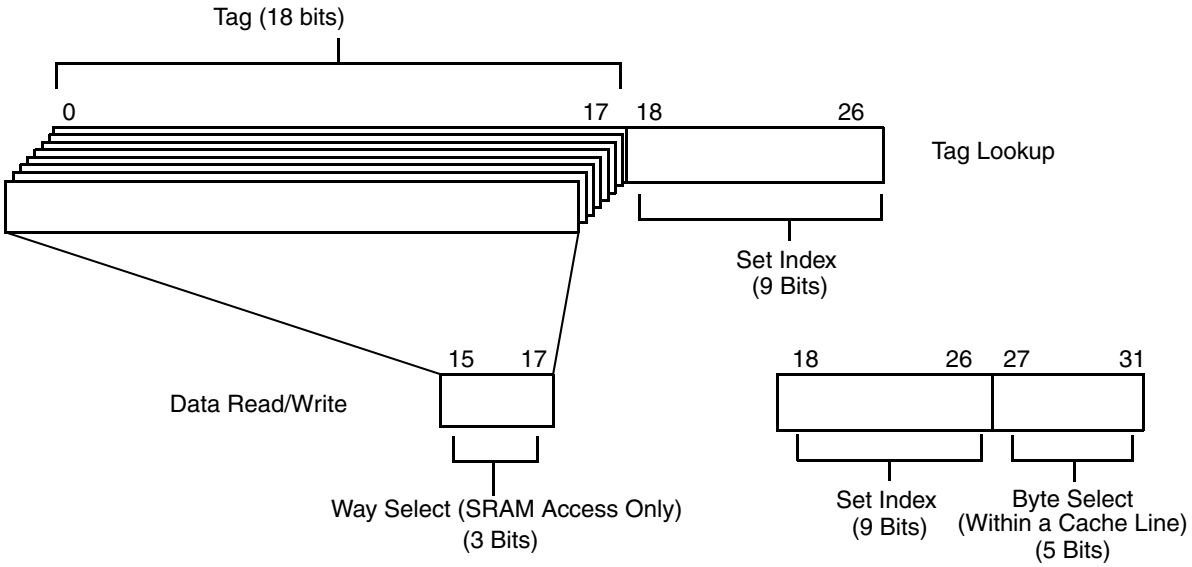


Figure 7-4. 128-Kbyte L2 Cache Address Configuration—Half SRAM, Half Cache Mode

The e500 core connects to the L2 cache and the system interface through the high-speed core complex bus (CCB). The e500 core and the L2 cache connect to the rest of the integrated device through the e500 coherency module (ECM). Figure 7-5 shows the data connections of the e500 core and L2/SRAM. The e500 core can simultaneously read 128 bits of data from the L2/SRAM, read 64 bits of data from the system interface, and write 128 bits of data to the L2/SRAM and/or system interface.

The L2/SRAM can be accessed by the e500 core or the system interface through the ECM. The L2 cache does not initiate transactions. Figure 7-5 shows the data bus connections of the e500 core and L2/SRAM.

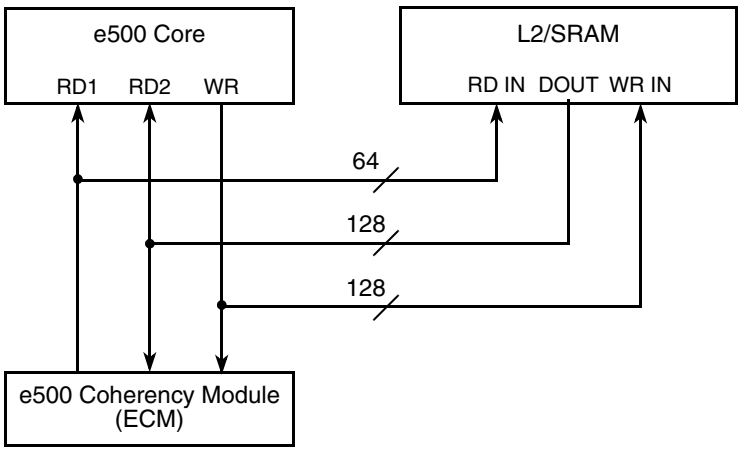


Figure 7-5. Data Bus Connection of CCB

Figure 7-6 shows address connections of the e500 core and L2/SRAM.

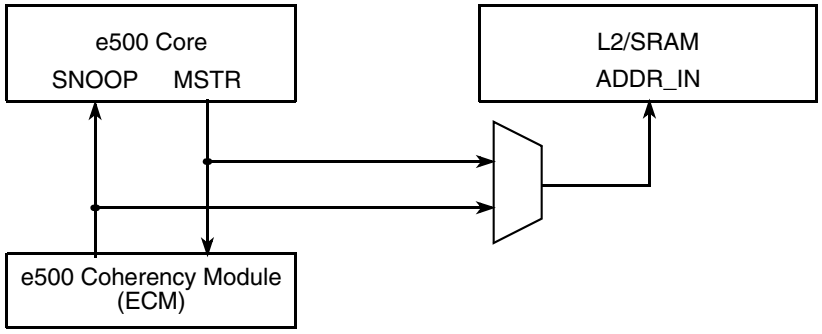


Figure 7-6. Address Bus Connection of CCB

In SRAM mode, if a non–cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non–cache-line write transaction cannot be allocated in the L2.

7.3 Memory Map/Register Definition

Table 7-1 shows the memory map for the L2/SRAM registers. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 7-1. L2/SRAM Memory-Mapped Registers

Offset	Register	Access	Reset	Section/Page
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	7.3.1.1/7-7
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E40	L2ERRDET—L2 error detect register	Special	0x0000_0000	7.3.1.5.2/7-14

Table 7-1. L2/SRAM Memory-Mapped Registers (continued)

Offset	Register	Access	Reset	Section/Page
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E50	L2ERRADDR—L2 error address capture register	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.5.2/7-14

7.3.1 L2/SRAM Register Descriptions

The following sections describe registers that control and configure the L2/SRAM array.

7.3.1.1 L2 Control Register (L2CTL)

The L2 control register (L2CTL), shown in Figure 7-7, controls configuration and operation of the L2/SRAM array. The sequence for modifying L2CTL is as follows:

1. **mbar**
2. **isync**
3. **stw** (WIMG = 01xx) CCSRBAR+0x2_0000
4. **lwz** (WIMG = 01xx) CCSRBAR+0x2_0000
5. **mbar**

	0	1	2	3	4	5	6		8	9	10	11	12	13	15
R	L2E	L2I	L2SIZ		L2BLKSZ		0	0	0	L2DO	L2IO	0	L2INTDIS	L2SRAM	
W															
Reset	0010_0000_0000_0000														

	16	17	18	19	20	21	22	23	24							31
R	0	0	L2LO	L2SLC	0	L2LFR	L2LFRID		0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x2_0000															

Figure 7-7. L2 Control Register (L2CTL)

Table 7-2 describes L2CTL fields.

Table 7-2. L2CTL Field Descriptions

Bits	Name	Description
0	L2E	L2 enable. Used to enable the L2 array (cache or memory-mapped SRAM). 0 The L2 SRAM (cache and memory-mapped SRAM) is disabled and is not accessed for reads, snoops, or writes. Setting the L2 Flash invalidate bit (L2I) is allowed. 1 The L2 SRAM (cache or memory-mapped SRAM) is enabled. Note that L2I can be set regardless of the value of L2E.
1	L2I	L2 Flash invalidate 0 The L2 status and LRU bits are not being cleared. 1 Setting L2I invalidates the L2 cache globally by clearing the all the L2 status bits, as well as the LRU algorithm. Memory-mapped SRAM is unaffected. Data in memory-mapped SRAM regions is unaffected by the Flash invalidate. The hardware automatically clears L2I when the invalidate is complete.
2–3	L2SIZ	L2 SRAM size (read only). Indicates the total available size of L2 SRAM (to be configured as cache or memory-mapped SRAM). 00 Reserved 01 Reserved 10 256 Kbyte 11 Reserved
4–5	L2BLKSZ	L2 cache/memory-mapped SRAM block size. Determines the L2 cache/memory-mapped SRAM block size. To change these bits, the L2 must be disabled (L2CTL[L2E] = 0). L2BLKSZ must be ≤ L2SIZ when L2E = 1. See the description of L2CTL[L2SRAM] for information on configuring the total SRAM as cache or memory-mapped SRAM. 00 Reserved 01 128 Kbyte 10 256 Kbyte 11 Reserved
6–8	—	Reserved
9	L2DO	L2 data-only. Reserved in full memory-mapped SRAM mode. L2DO may be changed while the L2 is enabled or disabled. 0 The L2 cache allocates entries for instruction fetches that miss in the L2. 1 The L2 cache allocates entries for processor data loads that miss in the L2 and for processor L1 castouts but does not allocate entries for instruction fetches that miss in the L2. Instruction accesses that hit in the L2, data accesses, and accesses from the system (including I/O stash writes) are unaffected. Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.
10	L2IO	L2 instruction-only. Reserved in full memory-mapped SRAM mode. Causes the L2 cache to allocate lines for instruction cache transactions only. L2IO may be changed while the L2 is enabled or disabled. 0 The L2 cache entries are allocated for data loads that miss in the L2 and for processor L1 castouts. 1 The L2 cache allocates entries for instruction fetch misses, but does not allocate entries for processor data transactions. Data accesses that hit in the L2, instruction accesses, and accesses from the system (including I/O stash writes) are unaffected. Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.
11	—	Reserved

Table 7-2. L2CTL Field Descriptions (continued)

Bits	Name	Description
12	L2INTDIS	Cache read intervention disable. Reserved for full memory-mapped SRAM mode. Used to disable cache read intervention. May be changed while the L2 is enabled or disabled. 0 Cache intervention is enabled. The ECM ensures that if a data read from another device hits in the L2 cache, it is serviced from the L2 cache. 1 Cache intervention is disabled
13–15	L2SRAM	L2 block assignment. Determines the L2 cache/memory-mapped SRAM block assignment. L2SIZ = L2BLKSIZ (1 block): 000 Block 0 = Cache 001 Block 0 = SRAM0 010–111 Reserved L2SIZ = L2BLKSIZ × 2 (2 blocks): Block 0 Block 1 000 Unused Cache 001 SRAM0 Unused 010 SRAM0 Cache 011 SRAM0 SRAM1 1xx Reserved To change these bits, the L2 must be disabled (L2CTL[L2E] = 0).
16–17	—	Reserved
18	L2LO	L2 cache lock overflow. Reserved in full memory-mapped SRAM mode. This sticky bit is set if an overlock condition is detected in the L2 cache. A lock overflow is triggered either by executing instruction or data cache block touch and lock set instructions or by performing L2 cache external writes with lock set. If all ways are locked and an attempt to stash is made, the stash is not allocated. 0 The L2 cache did not encounter a lock overflow. L2LO is cleared only by software. 1 The L2 cache encountered a lock overflow condition.
19	L2SLC	L2 snoop lock clear. This sticky bit is set if a snoop invalidated a locked data cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. L2SLC is reserved in full memory-mapped SRAM mode. 0 A snoop did not invalidate a locked L2 cache line. L2SLC is cleared only by software. 1 The L2 cache encountered a snoop that invalidated a locked line.
20	—	Reserved
21	L2LFR	L2 cache lock bits Flash reset. The L2 cache must be enabled (L2CTL[L2E] = 1) for reset to occur. This field is reserved in full memory-mapped SRAM mode. 0 The L2 cache lock bits are not cleared or the clear operation completed. 1 A reset operation is issued that clears each L2 cache line's lock bits. Depending on the L2LFRID value, data or instruction locks, or both can be reset. Cache access is blocked during this time. After L2LFR is set, the L2 cache unit automatically clears L2LFR when the reset operation is complete (if L2CTL[L2E] is set).
22–23	L2LFRID	L2 cache lock bits Flash reset select instruction or data. Indicates whether data or instruction lock bits or both are reset. 00 Not used 01 Reset data locks if L2LFR = 1 10 Reset instruction locks if L2LFR = 1 11 Reset both data and instruction locks if L2LFR = 1
24–31	—	Reserved

7.3.1.2 L2 Cache External Write Address Registers 0–3 (L2CEWAR*n*)

The MPC8555E supports allocating and locking of L2 cache lines from external agents, such as PCI. This functionality is called stashing. The L2 cache external write address registers 0–3 (L2CEWAR*n*) are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*) to control the cache external write functionality. Each register pair (for example, L2CEWAR0 and L2CEWCR0) specifies a programmed memory range that can be locked with a snoop write transaction. The address register must be naturally aligned to the window size in the corresponding control register. L2CEWAR*n* registers contain identical fields, as shown in Figure 7-8.

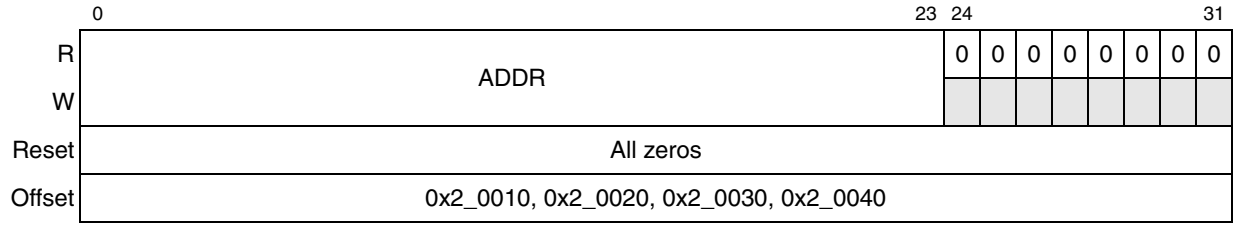


Figure 7-8. L2 Cache External Write Address Registers (L2CEWAR*n*)

Table 7-3 describes L2CEWAR*n* fields.

Table 7-3. L2CEWAR*n* Field Descriptions

Bits	Name	Description
0–23	ADDR	L2 cache external write base address
24–31	—	Reserved

7.3.1.3 L2 Cache External Write Control Registers 0–3 (L2CEWCR*n*)

The L2CEWAR*n* registers are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*), shown in Figure 7-9, to control cache external write functionality.

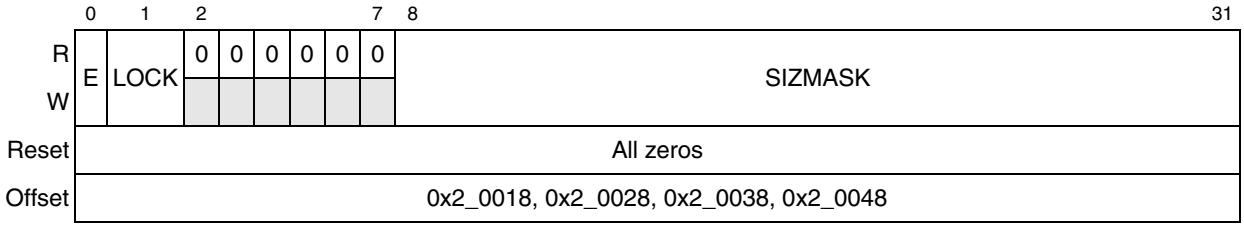


Figure 7-9. L2 Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

The L2CEWCR*n* registers contain identical fields, which are described in Table 7-4.

Table 7-4. L2CEWCR*n* Field Descriptions

Bits	Name	Description
0	E	External write enable. An external write matching the address window defined by L2CEWAR*n*/L2CEWCR*n* is allocated or updated in the L2 cache. 0 External writes for the L2CEWAR*n*/L2CEWCR*n* pair are disabled. 1 External writes are enabled for the L2CEWAR*n*/L2CEWCR*n* pair.
1	LOCK	Lock lines in the targeted cache. An external write matching the address window defined by L2CEWAR*n*/L2CEWCR*n* is locked in the L2 cache when it is allocated or updated. 0 The locked bit is not set when a line is allocated unless explicitly specified by transaction attributes. 1 Cache lines are allocated as locked. A hit to a valid, unlocked lines sets the lock.
2–7	—	Reserved
8–31	SIZMASK	Mask size. Defines the size of the naturally aligned address region for cache external writes. The address region must be aligned to a boundary that is a multiple of the mask size. Any value not listed below is illegal and produces boundedly undefined results. 1111 1111 1111 1111 1111 1111 256 bytes 1111 1111 1111 1111 1111 1110 512 bytes 1111 1111 1111 1111 1111 1100 1 Kbyte 1111 1111 1111 1111 1111 1000 2 Kbytes 1111 1111 1111 1111 1111 0000 4 Kbytes 1111 1111 1111 1111 1110 0000 8 Kbytes 1111 1111 1111 1111 1100 0000 16 Kbytes 1111 1111 1111 1111 1000 0000 32 Kbytes 1111 1111 1111 1111 0000 0000 64 Kbytes 1111 1111 1111 1110 0000 0000 128 Kbytes 1111 1111 1111 1100 0000 0000 256 Kbytes 1111 1111 1111 1000 0000 0000 512 Kbytes 1111 1111 1111 0000 0000 0000 1 Mbyte 1111 1111 1110 0000 0000 0000 2 Mbytes 1111 1111 1100 0000 0000 0000 4 Mbytes 1111 1111 1000 0000 0000 0000 8 Mbytes 1111 1111 0000 0000 0000 0000 16 Mbytes 1111 1110 0000 0000 0000 0000 32 Mbytes 1111 1100 0000 0000 0000 0000 64 Mbytes 1111 1000 0000 0000 0000 0000 128 Mbytes 1111 0000 0000 0000 0000 0000 256 Mbytes 1110 0000 0000 0000 0000 0000 512 Mbytes 1100 0000 0000 0000 0000 0000 1 Gbyte 1000 0000 0000 0000 0000 0000 2 Gbytes 0000 0000 0000 0000 0000 0000 4 Gbytes

7.3.1.4 L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBAR*n*)

The L2 memory-mapped SRAM base address registers (L2SRBAR*n*), shown in Figure 7-10, control the memory-mapped SRAM mode functionality. Specified addresses must be aligned to the value in L2CTL[L2BLKSZ]. If L2CTL[L2SRAM] specifies one memory-mapped SRAM block, its base address must be written to L2SRBAR0; if it specifies two memory-mapped SRAM blocks, L2SRBAR0 and L2SRBAR1 are used.

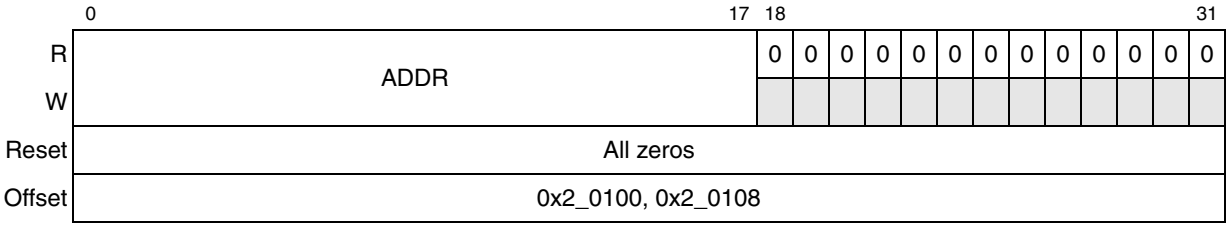


Figure 7-10. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)

L2SRBAR bits are described in Table 7-5.

Table 7-5. L2SRBAR*n* Field Descriptions

Bits	Name	Description
0–17	ADDR	L2 memory-mapped SRAM base address
18–31	—	Reserved

When enabled, the windows defined in L2SRBAR*n* supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR (see Section 4.3.1.1.2, "Configuration, Control, and Status Base Address Register (CCSRBAR)"). Overlapping SRAM and local access windows is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if SRAM and local access windows overlap. Furthermore, L2SRBARn should not overlap a DDR DRAM space for which a valid chip select is defined. This could result in spurious ECC errors if ECC and speculative reads are enabled. See Section 2.2.3.7, "Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects."

7.3.1.5 L2 Error Registers

L2 error detection, reporting, and injection allow flexible handling of ECC and parity errors in the L2 data and tag arrays. When the device detects an L2 error, the appropriate bit in the error detect register (L2ERRDET) is set. Error detection is disabled by setting the corresponding bit in the error disable register (L2ERRDIS).

The address and attributes of the first detected error are also saved in the error capture registers (L2ERRADDR, L2ERRATTR, L2CAPTDATAHI, L2CAPTDATALO, and L2CAPTACC). Subsequent errors set error bits in the error detection registers, but information is saved only for the first one. Error reporting (by generating an interrupt) is enabled by setting the corresponding bit in the error interrupt enable register (L2ERRINTEN). Note that the error detect bit is set regardless of the state of the interrupt enable bit. When an error is detected, if error detection is enabled the L2 cache/SRAM always asserts an internal error signal with read data to prevent the L1 caches and architectural registers from being loaded with corrupt data. If error detection is disabled, the detected error bit is not set and no internal signal is asserted.

The L2 error detect register (L2ERRDET) is implemented as a bit-reset type register. Reading from this register occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0x0200_0000 is written to the register.

Note that in SRAM mode, if a non–cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non–cache-line write transaction cannot be allocated in the L2.

7.3.1.5.1 Error Injection Registers

The L2 cache includes support for injecting errors into the L2 data, data ECC, or tag. This may be used to test error recovery software by deterministically creating error scenarios.

The preferred method for error injection is to set all data pages to cache-inhibited (MMU TLB entry I = 1) except a scratch page, set L2CTL[L2DO] to prevent allocation of instruction accesses, and invalidate the L2 by setting L2CTL[L2I] = 1. The following code sequence triggers an error, then detects it (A is an address in the scratch page):

```
dcbz A          | allocates the line in the L1 in the modified state
dcbtls_L2 A     | forces the line from the L1 and allocates the line in the L2
lwz A
```

Data or tag errors are injected into the line, according to the error injection settings in L2ERRINJHI, L2ERRINJLO, and L2ERRINJCTL, at allocation. The final load detects and reports the error (if enabled) and allows software to examine the offending data, address, and attributes.

Note that error injection enable bits in L2ERRINJCTL must be cleared by software and the L2 must be invalidated (by setting L2CTL[L2I]) before resuming L2 normal operation. Figure 7-11 shows the L2 error injection mask high register (L2ERRINJHI).

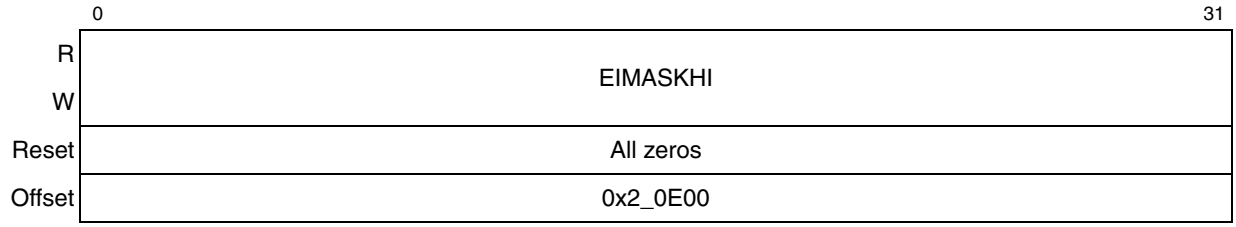


Figure 7-11. L2 Error Injection Mask High Register (L2ERRINJHI)

Table 7-6 describes L2ERRINJHI[EIMASKHI].

Table 7-6. L2ERRINJHI Field Descriptions

Bits	Name	Description
0–31	EIMASKHI	Error injection mask/high word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on cache/SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

Figure 7-12 shows the L2 error injection mask low register (L2ERRINJLO) fields.

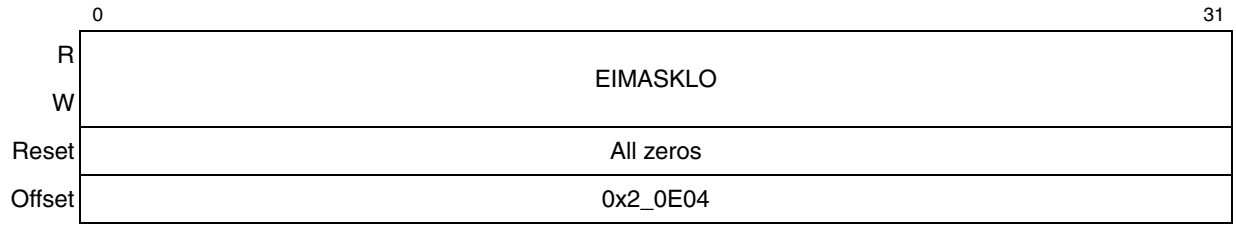


Figure 7-12. L2 Error Injection Mask Low Register (L2ERRINJLO)

Table 7-7 describes L2ERRINJLO[EIMASKLO].

Table 7-7. L2ERRINJLO Field Descriptions

Bits	Name	Description
0–31	EIMASKLO	Error injection mask/low word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

Figure 7-13 shows the L2 error injection mask control register (L2ERRINJCTL).

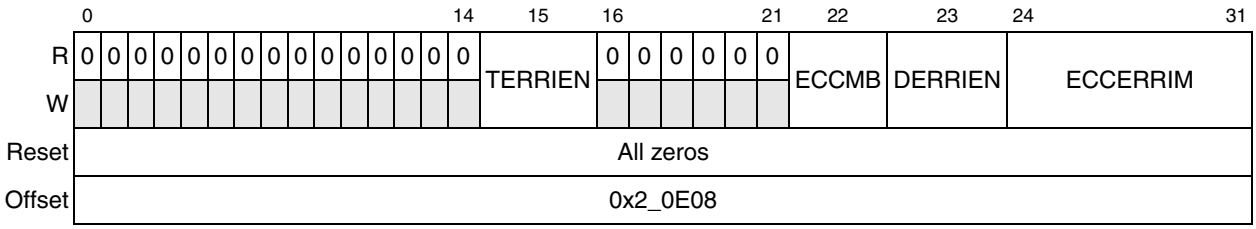


Figure 7-13. L2 Error Injection Mask Control Register (L2ERRINJCTL)

Table 7-8 describes L2ERRINJCTL fields.

Table 7-8. L2ERRINJCTL Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	TERRIEN	L2 tag array error injection enable 0 No tag errors are injected. 1 All subsequent entries written to the L2 tag array have the parity bit inverted.
16–21	—	Reserved
22	ECCMB	ECC mirror byte enable 0 ECC byte mirroring is disabled 1 The most significant data path byte is mirrored onto the ECC byte if DERRIEN = 1.
23	DERRIEN	L2 data array error injection enable 0 No data errors are injected. 1 Subsequent entries written to the L2 data array have data or ECC bits inverted as specified in the data and ECC error injection masks and/or data path byte mirrored onto ECC as specified by ECC mirror byte enable. **Note:** If both ECC mirror byte and data error injection are enabled, ECC mask error injection is performed on the mirrored ECC.
24–31	ECCERRIM	Error injection mask for the ECC bits. A set bit corresponding to an ECC bit causes that bit to be inverted on SRAM writes if DERRIEN = 1.

7.3.1.5.2 Error Control and Capture Registers

The error control and capture registers control detection and reporting of tag parity, ECC and L2 configuration errors. L2 configuration errors are illegal combinations of L2 size and block size and are

detected when the L2 is enabled (L2CTL[L2E] = 1). Figure 7-14 shows the L2 error capture data high register (L2CAPTDATAHI).

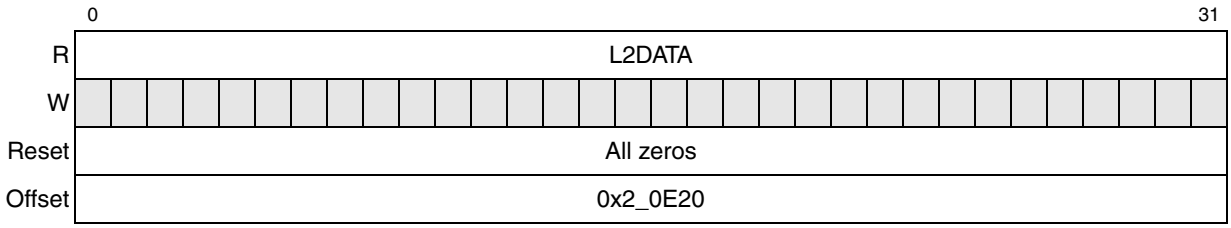


Figure 7-14. L2 Error Capture Data High Register (L2CAPTDATAHI)

Table 7-9 describes L2CAPTDATAHI[L2DATA] fields.

Table 7-9. L2CAPTDATAHI Field Descriptions

Bits	Name	Description
0–31	L2DATA	L2 data high word

Figure 7-15 shows the L2 error capture data low register (L2CAPTDATALO).

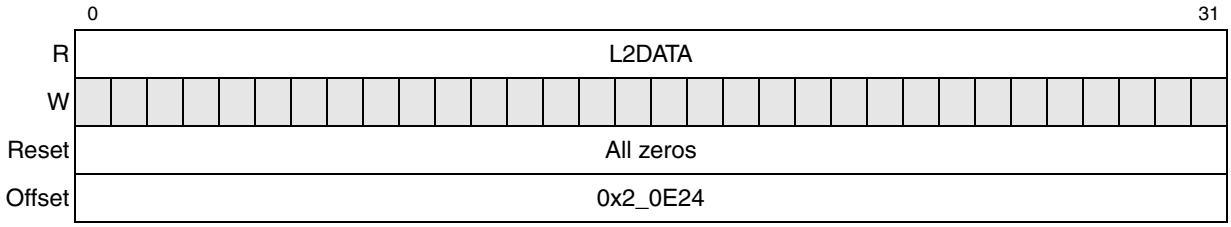


Figure 7-15. L2 Error Capture Data Low Register (L2CAPTDATALO)

Table 7-10 describes L2CAPTDATALO[L2DATA] fields.

Table 7-10. L2CAPTDATALO Field Descriptions

Bits	Name	Description
0–31	L2DATA	L2 data low word

Figure 7-16 shows the L2 error syndrome register (L2CAPTECC).

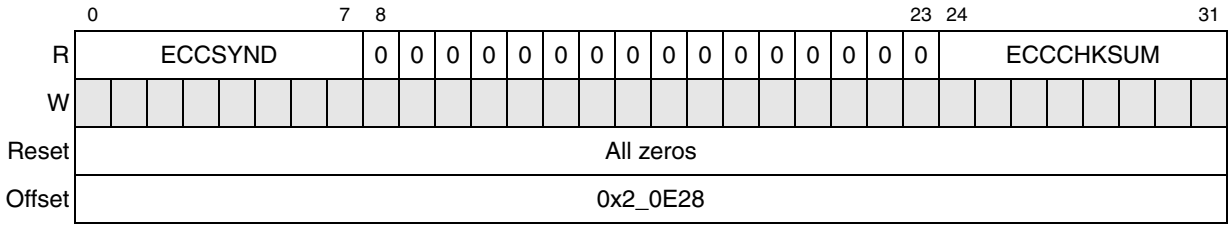


Figure 7-16. L2 Error Syndrome Register (L2CAPTECC)

Table 7-11 describes L2CAPTECC fields.

Table 7-11. L2CAPTECC Field Descriptions

Bits	Name	Description
0–7	ECCSYND	The calculated ECC syndrome of the failing double word
8–23	—	Reserved
24–31	ECCCHKSUM	The data path ECC of the failing double word

Figure 7-17 shows the L2 error detect register (L2ERRDET).

	0	1–26	27	28	29	30	31
R	MULL2ERR	0	TPARERR	MBECCERR	SBECCERR	0	L2CFGDIS
W							
Reset	0000_0000_0000_0000_0000_0000_0000_0000						
Offset	0x2_0E40						

Figure 7-17. L2 Error Detect Register (L2ERRDET)

Table 7-12 describes L2ERRDET fields.

Table 7-12. L2ERRDET Field Descriptions

Bits	Name	Description
0	MULL2ERR	Multiple L2 errors (bit reset, write 1 to clear) 0 Multiple L2 errors of the same type were not detected. 1 Multiple L2 errors of the same type were detected.
1–26	—	Reserved
27	TPARERR	Tag parity error (bit reset, write 1 to clear) 0 Tag parity error was not detected 1 Tag parity error was detected Note that if an L2 cache tag parity error occurs on an attempt to write a new line, the L2 cache must be Flash invalidated. L2 functionality is not guaranteed if Flash invalidation is not performed after a tag parity error.
28	MBECCERR	Multiple-bit ECC error (bit reset, write 1 to clear) 0 Multiple-bit ECC errors were not detected. 1 Multiple-bit ECC errors were detected.
29	SBECCERR	Single-bit ECC error (bit reset, write 1 to clear) 0 Single-bit ECC error was not detected 1 Single-bit ECC error was detected
30	—	Reserved
31	L2CFGERR	L2 configuration error (bit reset, write 1 to clear). Reports inconsistencies between the L2SIZ, L2BLKSZ, and L2SRAM settings of the L2 control register (L2CTL). 0 L2 configuration errors were not detected 1 L2 illegal configuration error detected

Figure 7-18 shows the L2 error disable register (L2ERRDIS).

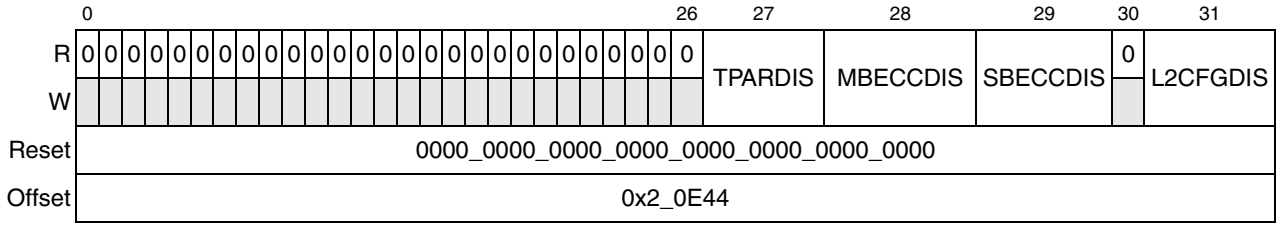


Figure 7-18. L2 Error Disable Register (L2ERRDIS)

Table 7-13 describes L2ERRDIS fields.

Table 7-13. L2ERRDIS Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARDIS	Tag parity error disable 0 Tag parity error detection enabled 1 Tag parity error detection disabled
28	MBECCDIS	Multiple-bit ECC error disable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBECCDIS must be cleared and L2ERRINTEN[MBECCINTEN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Multiple-bit ECC error detection enabled 1 Multiple-bit ECC error detection disabled
29	SBECCDIS	Single-bit ECC error disable 0 Single-bit ECC error detection enabled 1 Single-bit ECC error detection disabled
30	—	Reserved
31	L2CFGDIS	L2 configuration error disable 0 L2 configuration error detection enabled 1 L2 configuration error detection disabled

Figure 7-19 shows the L2 error interrupt enable register (L2ERRINTEN). When an enabled error condition exists, the L2 signals an interrupt to the core through the internal *int* signal.

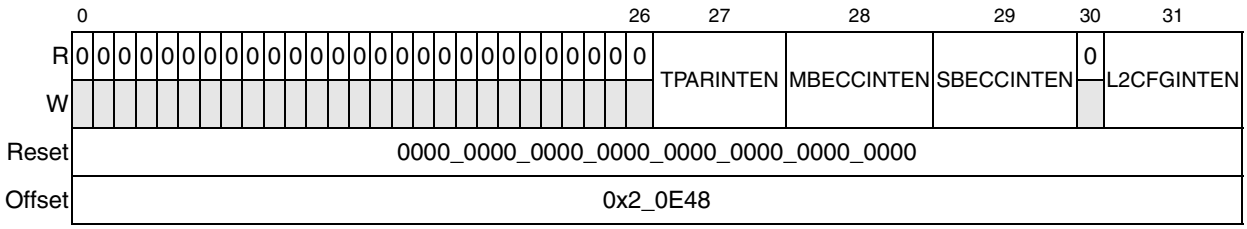


Figure 7-19. L2 Error Interrupt Enable Register (L2ERRINTEN)

Table 7-14 describes L2ERRINTEN fields.

Table 7-14. L2ERRINTEN Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARINTEN	Tag parity error reporting enable 0 Tag parity error reporting disabled 1 Tag parity error reporting enabled
28	MBECCINTEN	Multiple-bit ECC error reporting enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, L2ERRDIS[MBECCDIS] must be cleared and MBECCINTEN must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Multiple-bit ECC error reporting disabled 1 Multiple-bit ECC error reporting enabled
29	SBECCINTEN	Single-bit ECC error reporting enable 0 Single-bit ECC error reporting disabled 1 Single-bit ECC error reporting enabled
30	—	Reserved
31	L2CFGINTEN	L2 configuration error reporting enable 0 L2 configuration error reporting disabled 1 L2 configuration error reporting enabled

Figure 7-20 shows the L2 error attributes capture register (L2ERRATTR).

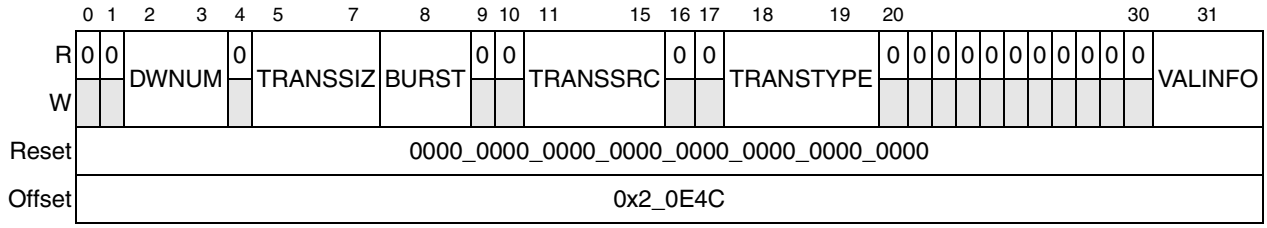


Figure 7-20. L2 Error Attributes Capture Register (L2ERRATTR)

Table 7-15 describes L2ERRATTR fields.

Table 7-15. L2ERRATTR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–3	DWNUM	Double-word number of the detected error (data ECC errors only)
4	—	Reserved
5–7	TRANSSIZ	Transaction size for detected error Single-beat / Burst / Single-beat / Burst 000 8 bytes / Reserved / 100 4 bytes / Reserved 001 1 byte / 16 bytes / 101 5 bytes / Reserved 010 2 bytes / 32 bytes / 110 6 bytes / Reserved 011 3 bytes / Reserved / 111 7 bytes / Reserved

Table 7-15. L2ERRATTR Field Descriptions (continued)

Bits	Name	Description
8	BURST	Burst transaction for detected error 0 Single-beat (≤ 64 bits) transaction 1 Burst transaction
9–10	—	Reserved
11–15	TRANSSRC	Transaction source for detected error 00000 External (system logic) 10000 Processor (instruction) 10001 Processor (data)
16–17	—	Reserved
18–19	TRANSTYPE	Transaction type for detected error 00 Snoop (tag/status read) 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VALINFO	L2 capture registers valid 0 L2 capture registers contain no valid information or no enabled errors were detected. 1 L2 capture registers contain information of the first detected error which has reporting enabled. Software must clear this bit to unfreeze error capture so error detection hardware can overwrite the capture address/data/attributes for a newly detected error.

Figure 7-21 shows the L2 error address capture register (L2ERRADDR).

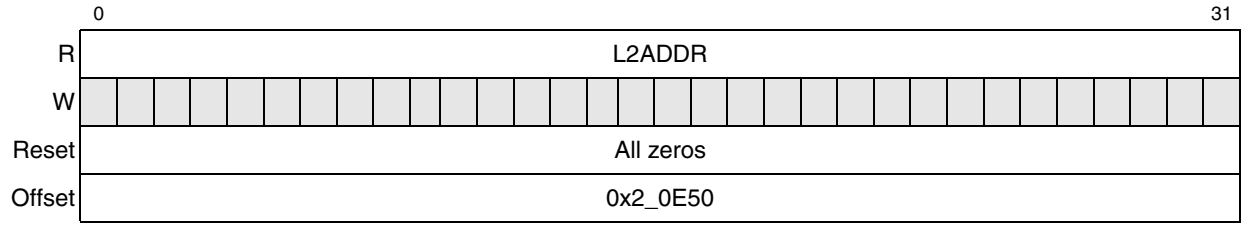


Figure 7-21. L2 Error Address Capture Register (L2ERRADDR)

Table 7-16 describes L2ERRADDR[L2ADDR] fields.

Table 7-16. L2ERRADDR Field Descriptions

Bits	Name	Description
0–31	L2ADDR	L2 address corresponding to detected error

Figure 7-22 shows the L2 error control register (L2ERRCTL).

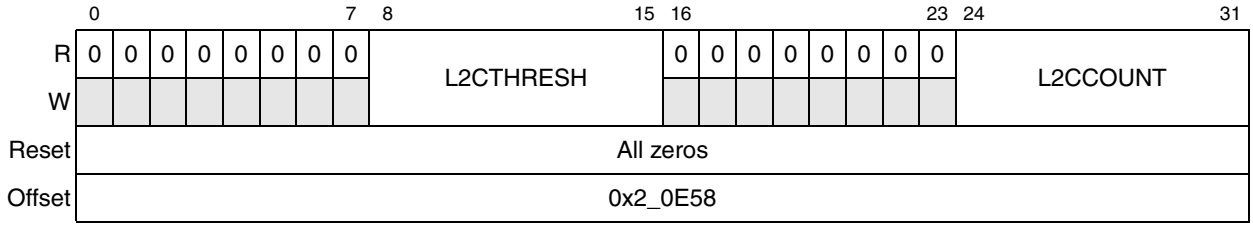


Figure 7-22. L2 Error Control Register (L2ERRCTL)

Table 7-17 describes L2ERRCTL fields.

Table 7-17. L2ERRCTL Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	L2CTHRESH	L2 cache threshold. Threshold value for the number of ECC single-bit errors that are detected before reporting an error condition
16–23	—	Reserved
24–31	L2CCOUNT	L2 count. Counts ECC single-bit errors detected. If L2CCOUNT equals the ECC single-bit error trigger threshold, an error is reported if single-bit error reporting is enabled

7.4 External Writes to the L2 Cache (Cache Stashing)

Data from an I/O master can be allocated into the L2 cache while simultaneously being written to memory. External writes can be performed from any I/O master:

- Ethernet
- CPM
- PCI
- DMA

New cache lines are allocated for full cache line writes, unless the line is already valid or locked. Sub–cache-line write data is stashed only if the line is valid in the cache. A read-modify-write process is used to merge the stashed data with the valid line data.

The L2 cache external write address registers 0–3 (L2CEWAR*n*) are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*) to control the cache stashing functionality. Each register pair (for example, L2CEWAR0 and L2CEWCR0) specifies a programmed memory range that can be locked with a snoop write transaction. The address register must be naturally aligned to the window size in the corresponding control register. For more information, see Section 7.3.1.2, “L2 Cache External Write Address Registers 0–3 (L2CEWARn),” and Section 7.3.1.3, “L2 Cache External Write Control Registers 0–3 (L2CEWCRn).”

Note that stashing can occur regardless of whether the L1 cache is enabled.

Note that stashing can also occur even if the cache-inhibited bit in the MMU is set for the page. When the core looks for (and does not find) the stashed data in the L1 cache, a transaction is generated in which the L2 cache responds by updating the L1 cache with the requested data.

For information on how to initiate cache stashing, see the respective chapters for the I/O masters, listed above, that support stashing.

7.5 L2 Cache Timing

Table 7-18 shows the timing of back-to-back loads that miss in the L1 data cache and hit in the L2 cache, assuming the core is running at 2 1/2 times the L2 cache frequency. The L2 returns the 128 bits containing the requested data (critical quad word) first. This data is forwarded to the result register before the full cache line reloads the L1.

Table 7-18. Fastest Read Timing—Hit in L2

Core Clocks	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
e500 core load 1	To D-cache	D-cache miss	To CIU	CIU Q												To CIU	LSU DLFB	LSU reads command	LSU reads out data	Result bus						
e500 core load 2		to D-cache	D-cache miss	to CIU	CIU Q																To CIU	LSU DLFB	LSU reads command	LSU reads out data	Result bus	
CCB clocks	<1	2		3		4		5		6		7		8		9		10		11>						
CCB address bus load 1		BG		TS				AACK		HIT DATA-COMING																
CCB address bus load 2						BG		TS				AACK		HIT DATA-COMING												
CCB data bus load 1												DATA		DATA												
CCB data bus load 2																				DATA			DATA			

7.6 L2 Cache and SRAM Coherency

This section explains the rules of cache and memory-mapped SRAM coherency. The term ‘snoop transaction’ refers to transactions initiated by the device system logic or by I/O traffic, as opposed to e500 core-initiated transactions.

7.6.1 L2 Cache Coherency Rules

L2 cache coherency rules are as follows:

- The L2 is non-inclusive of the L1—valid L1 lines may be valid or invalid in the L2.
- The L2 cache holds no modified data. There are four states—invalid, exclusive, exclusive locked, and stale.
- The L2 allocates entries for data cast-out or pushed (non-global, non-write-through write with kill) from the L1 caches.
- Lines for e500 core-initiated burst read transactions are allocated as exclusive in the L2.
- The L2 supports I/O devices reading data from valid lines in the L2 cache (data intervention) if L2CTL[L2INTDIS] = 0. An optional unlock attribute causes I/O reads to clear a lock when the read is performed.
- The L2 cache does not respond to cache-inhibited read transactions.
- e500 core-initiated, cache-inhibited store transactions invalidate the line when they hit on a valid L2 line. If the line is locked, it goes to the stale state. For other write transactions the cache-inhibited bit is ignored.
- Non-burst cacheable write transactions from the e500 core (generated by write-through cacheable stores) update a valid L2 cache line through a read-modify-write operation.
- e500 core cast out transactions that hit on a stale line in the L2 cache cause a data update of the line and a change to the valid locked state for that line.
- An e500 core-initiated, cacheable, non-write-through store that misses in L1 and hits on a line in the L2 invalidates that line in the L2. If the line is marked exclusive locked, the L2 marks the line as stale.
- Transactions that hit a stale L2 cache line that would cause an allocate if they miss cause a data update of the line (when data arrives from memory) and a change to the line’s valid locked state. Data is not supplied by the L2 cache for the read in this case.
- The following transactions kill the data and the respective locks when they hit a valid L2 line:
  - **dcbf**
  - **dcbi**
- The L2 cache supports mixed cache external writes and core-initiated writes to the same addresses if the core-initiated writes are marked coherency-required, caching allowed, not write-through (WIMG = 001x) and the external writes are marked coherency-required, caching-allowed.
- The L2 cache supports writes to the L2 cache from peripheral devices or from I/O controllers through snoop write transactions with addresses that hit in a programmed memory range. Full-cache-line (32-byte) write transactions update the data for a valid line in the L2, and if the line

is not valid in the L2, a line is allocated. Sub-cache-line write transactions update the data only for valid L2 cache lines through read-modify-write operations.

- The L2 cache supports burst writes that lock an L2 cache line from peripheral devices or from I/O controllers through write transactions with addresses that hit in a programmed memory range that has the lock attribute set.
- The L2 cache supports burst writes that allocate and/or lock an L2 cache line from peripheral devices or I/O controllers through a write allocate transaction. See the system logic programming model (for example, that of the DMA controller) for details on how to set the transaction type for cache external writes to the L2.

7.6.2 Memory-Mapped SRAM Coherency Rules

Memory-mapped SRAM coherency rules are as follows:

- External (non–core-initiated) accesses to memory-mapped SRAM must be marked coherency-required. External accesses marked coherency-not-required to memory-mapped SRAM may cause an address unavailable error.
- Accesses to memory-mapped SRAM are cacheable only in the corresponding e500 L1 caches. External accesses must be marked cache-inhibited or be performed with non-caching transactions.

7.7 L2 Cache Locking

The device caches can be locked and cleared using the following methods:

- Cache locking methods
  - Individual line locks are set and cleared by using instructions defined by the architecture, which is part of the Freescale embedded category implementation standards (EIS). These instructions include Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**). For detailed information about these instructions, see the *PowerPC™ e500 Core Family Reference Manual*.
  - A lock attribute can be attached to write operations.
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWAR*n*).
  - The entire cache can be locked by setting configuration registers appropriately.
- Methods for clearing locks
  - Individual locks are cleared by cache locking instructions (Instruction Cache Block Lock Clear (**icblc**) and Data Cache Block Lock Clear (**dcblc**)) or by snooped flush unless the entire cache is locked.
  - Flash clearing of all instruction and/or data locks can be done by writes to configuration registers.
  - An unlock attribute can be attached to I/O read operations.

7.7.1 Locking the Entire L2 Cache

The entire L2 cache can be locked by setting L2CTL[L2DO] = 1 and L2CTL[L2IO] = 1. This has the effect of preventing any further allocation of new lines in the cache by core requests. If there are lines in the cache that are not valid, they cannot be used by core requests until the cache is unlocked. While the cache is locked, read requests are serviced as normal, and snooping continues as normal to maintain coherency. Lines invalidated to satisfy coherency requirements cannot be reallocated by core requests while the cache remains locked. The L2 cache can be unlocked by clearing L2CTL[L2IO] and/or L2CTL[L2DO]. Note that L2CTL[L2DO] and L2CTL[L2IO] have no effect on cache external write allocations or memory-mapped SRAM.

Note that this form of cache locking does not use the lock bits of the cache and cannot be cleared by resetting the cache or lock bits.

7.7.2 Locking Programmed Memory Ranges

A programmed memory range can be locked with a snoop write transaction that matches a cache external write address range (specified by L2CEWAR*n* and L2CEWCR*n*). There is no clearing of locks through the programmed address ranges. Locks can be cleared using clear lock instructions, flushes, or read-and-clear-lock snoop (RWNITC with clear lock attribute), or Flash clear locks.

7.7.3 Locking Selected Lines

Individual lines are locked when the L2 receives one of the following burst transactions:

- **icbtls** (CT = 1)—Instruction Cache Block Touch and Lock Set instruction
- **dcbtls** (CT = 1)—Data Cache Block Touch and Lock Set instruction
- **dcbtstls** (CT = 1)—Data Cache Block Touch for Store and Lock Set instruction
- Snoop burst write—If the address hits on a programmed cache external write space with the lock attribute set, or if the write allocate transaction type is used
- Snoop non-burst write—If the address hits on a programmed cache external write space with the lock attribute set

Note that the core complex broadcasts these instructions to the L2 if the CT field in the instruction specifies the L2 cache (CT = 1). When the L2 cache is specified, data is not placed in the L1, only the L2. If the L1 cache is specified (CT = 0), the L2 does not lock the line, and the data is placed in the L1 (and locked).

When the touch lock set L2 instruction (**dcbtls** or **dcbtstls**) hits are modified in the L1 cache, the modified data is allocated into the L2 cache (and written back to main memory) and a data lock is set. The L1 line state transitions to invalid.

Note that if the L2 receives a request to allocate and lock a line, but all lines in the selected way are locked, the requested L2 line is not allocated and the L2 cache lock overflow bit (L2CTL[L2LO]) is set.

Lines invalidated to satisfy coherency requirements cannot be reallocated while the cache remains locked.

7.7.4 Clearing Locks on Selected Lines

Individual locks in the L2 are cleared by a lock clear (**icblc** or **dcblc,** CT = 1) instruction. This directs the L2 cache to clear a lock on that line if it hits in the L2 cache. Both data and instruction locks are cleared by the **icblc** and **dcblc** instructions.

Note that the lock on a line is cleared if the line is invalidated by a snooped flush transaction, and the line in the cache is available for allocation of a new line of instruction or data unless the entire cache is locked.

NOTE

There is a scenario in which a lock clear operation appears to fail to clear a lock in the L2 cache. This occurs only when the attempt to set the lock results in a bus error (for example, PCI returns an error condition).

Assume the following scenario:

1. The e500 attempts to set a lock in the L2 cache (by executing a **dcbtls** or **icbtls** instruction with CT = 1). The line is not already present in the cache, so it must be read from external memory. This read encounters an error which, depending on the chip configuration, will be reported to the core (probably as an interrupt).
2. At (or near) the same time, a cache external write to the same cache line is being mastered by the ECM.
3. Very soon after the cache external write, a transaction to clear the lock occurs. This can be caused by the processor executing a **dcblc** or **icblc** instruction with CT = 1, or by the ECM mastering a lock clear transaction.

If this scenario occurs within a tight timing window, the cache line may unexpectedly remain locked at the end of the sequence.

The interrupt handler may want to clear the erroneously remaining lock in this case.

7.7.5 Flash Clearing of Instruction and Data Locks

Locks for instructions and data are recorded separately in the L2 cache, and they can be Flash cleared separately by writing the appropriate value to the L2 cache control register (L2CTL[L2LFR] and L2CTL[L2LFRID]). Flash invalidating of the L2 (setting L2CTL[L2I]) clears all locks on both instructions and data.

Note that Flash clearing is the only way to clear data locks without clearing instruction locks, or to clear instruction locks without clearing data locks. All instructions and snoop transactions that clear locks clear both data and instruction locks.

7.7.6 Locks with Stale Data

If data is locked in the L2 and either the e500 core performs a cacheable copyback store or a **dcbtst** misses in the L1, the L2 invalidates the line; however, the L2 clears the valid bit for the data, the lock remains, and the line cannot be victimized. If the e500 core casts out modified data or pushes it in response to a non-flush snoop, the L2 updates the data and sets the valid bit again, maintaining the lock and keeping the data in the cache hierarchy.

7.8 PLRU L2 Replacement Policy

Line replacement is determined using a pseudo least-recently-used (PLRU) algorithm. There is a valid bit (V0–V7) for each line. To determine the replacement victim (the line to be cast out), there are seven PLRU bits (P0–P6) for each set. PLRU bits are updated every time a new line is allocated or replaced and every time a line is modified or invalidated. There are two sets of lock bits, one for instructions (I0–I7) and one for data (D0–D7) for every line. The lock bits act as a mask over the PLRU bits to determine victim selection. The PLRU bits are updated regardless of line locking.

Figure 7-23 shows the binary decision tree used to generate the victim line. The eight ways of the L2 cache are labeled W0–W7; the seven PLRU bits are labeled P0–P6.

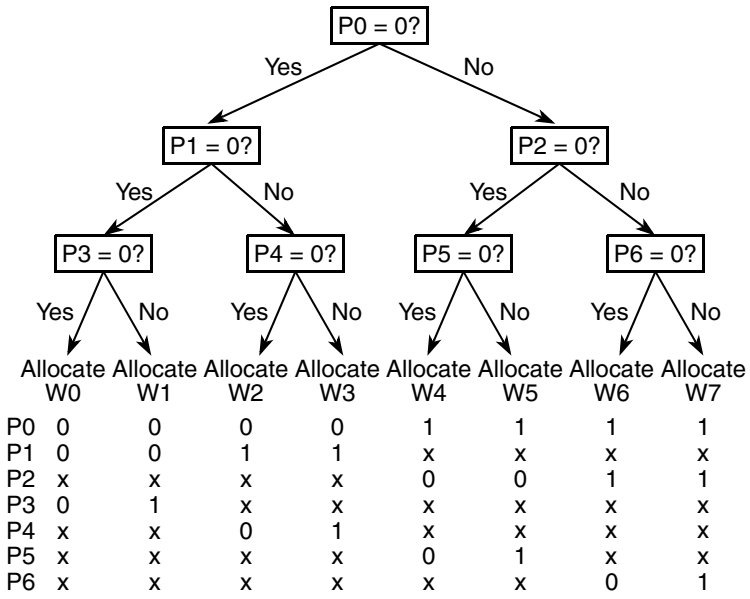


Figure 7-23. L2 Cache Line Replacement Algorithm

7.8.1 PLRU Bit Update Considerations

PLRU bits are updated when a way is modified, either by a write, an invalidate, or an allocation of a new line. PLRU bit updates depend on which cache way was last accessed, as summarized in Table 7-19.

Table 7-19. PLRU Bit Update Algorithm

Last Way Accessed	PLRU Bits						
	P0	P1	P2	P3	P4	P5	P6
0	1	1	—	1	—	—	—
1	1	1	—	0	—	—	—
2	1	0	—	—	1	—	—
3	1	0	—	—	0	—	—
4	0	—	1	—	—	1	—
5	0	—	1	—	—	0	—
6	0	—	0	—	—	—	1
7	0	—	0	—	—	—	0

When an L2 line is invalidated, the PLRU bits are updated, marking the corresponding way as least-recently-used. This causes the invalidated way to be selected as the next victim.

7.8.2 Allocation of Lines

L2 cache lines are locked through the status array lock bits. There are two lock bits for each way of each set (1024 sets by eight ways). These bits are set or cleared through special L2 controller commands.

Lock bits are used at allocate time to steer the PLRU algorithm away from selecting locked victims. In the following discussion, the eight lock bits for a particular set are called L0–L7.

```
Where Lock Way i: Li = Di | Ii, i=0...7 (Di = data lock, Ii = instruction lock)
```

An effective value of each PLRU bit is calculated as follows:

```
P0_eff = f(P0,L0,L1,L2,L3,L4,L5,L6,L7) = (L0 & L1 & L2 & L3) | (P0 & ~(L4 & L5 & L6 & L7))
P1_eff = f(P1,L0,L1,L2,L3) = (L0 & L1) | (P1 & ~(L2 & L3))
P2_eff = f(P2,L4,L5,L6,L7) = (L4 & L5) | (P2 & ~(L6 & L7))
P3_eff = f(P3,L0,L1) = L0 | (P3 & ~L1)
P4_eff = f(P4,L2,L3) = L2 | (P4 & ~L3)
P5_eff = f(P5,L4,L5) = L4 | (P5 & ~L5)
P6_eff = f(P6,L6,L7) = L6 | (P6 & ~L7)
```

These effective PLRU bits are used to select a victim, as indicated in Table 7-20.

Table 7-20. PLRU-Based Victim Selection Mechanism

Way Selected	PLRU State (Binary)	Reduced Logic Equation
W0	00x0xxx	~P0 & ~P1 & ~P3
W1	00x1xxx	~P0 & ~P1 & P3
W2	01xx0xx	~P0 & P1 & ~P4
W3	01xx1xx	~P0 & P1 & P4
W4	1x0xx0x	P0 & ~P2 & ~P5
W5	1x0xx1x	P0 & ~P2 & P5
W6	1x1xxx0	P0 & P2 & ~P6
W7	1x1xxx1	P0 & P2 & P6

7.9 L2 Cache Operation

This section describes the behavior of the L1 and L2 cache in response to various operations and in various configurations.

7.9.1 L2 Cache States

The L2 status array uses four bits for each line to determine the status of the line. Different combinations of these bits result in different L2 states. The status bits are as follows:

- Valid (V)
- Instruction locked (IL)
- Data locked (DL)
- Stale (T)

Table 7-21 shows L2 cache states. Note that these conventions are also used in Table 7-22.

Table 7-21. L2 Cache States

V	T	IL	DL	L2 states
0	x	x	x	Invalid (I)
1	0	0	0	Exclusive (E)
1	0	0	1	Exclusive data locked (EDL)
1	0	1	0	Exclusive instruction locked (EIL)
1	0	1	1	Exclusive instruction and data locked (EL)
1	1	0	0	Stale (data invalid, locks invalid) (T)
1	1	0	1	Stale (data invalid, dlock valid) (TDL)
1	1	1	0	Stale (data invalid, ilock valid) (TIL)
1	1	1	1	Stale (data invalid, locks valid) (TL)

7.9.2 Flash Invalidation of the L2 Cache

The L2 cache may be completely invalidated by setting the L2I bit of the L2 control register (L2CTL). Note that no data is lost in this process because the L2 cache is a write-through cache and contains no modified data. Flash invalidation of the cache is necessary when the cache is initially enabled and may be necessary to recover from some error conditions such as a tag parity error.

The invalidation process requires several cycles to complete. The L2I bit remains set during this procedure and is then cleared automatically when the procedure is complete. The L2 cache controller issues retries for all transactions on the e500 core complex bus (CCB) while the Flash invalidation process is in progress.

Note that the contents of memory-mapped SRAM regions of the data array are unaffected by a Flash invalidation of the L2 cache regions of the array.

7.9.3 L2 State Transitions

Table 7-22 lists state transitions for all e500 core-initiated transactions that change the L2 cache state. Core-initiated transactions caused when the core executes **msync**, **mbar**, **tlbivax**, or **tlbsync** do not change the L2 cache state. The table does not list initial L1 states for transactions that hit in the L1 (iL1 or dL1) and are not sent to the L2.

In the table, the heading ‘L2 hit’ indicates that the L2 provides (on a read) or captures (on a write) data for an existing line. Some entries list two final L1 states. L2 touch instructions never allocate into iL1 or dL1.

Note that if the L2 SRAM is disabled, the L2 initial and final states are always I and the L2 never hits. Similarly, if the L2 SRAM is in full memory-mapped SRAM mode, the L2 initial and final states are always I and the L2 never hits for addresses not in the memory-mapped SRAM address range. The L2 always hits for addresses in the enabled memory-mapped SRAM address ranges.

Table 7-22. State Transitions Due to Core-Initiated Transactions

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cacheable instruction fetch icbtls_L1	iL1 I	I/T	No	I/V	Same	L2CTL[L2DO] = 1. L2 touch instructions not allocated in L1
		I	No	I/V	E	L2CTL[L2DO] = 0
icbt_L2	dL1 I,E	E/EL	Yes	I/V	Same	
		T	No	I/V	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus
icbtls_L2	dL1 I,E	I/T	No	I	Same	L2CTL[L2DO] = 1
		E	Yes	I	I	L2CTL[L2DO] = 1
		EL	Yes	I	T	L2CTL[L2DO] = 1
		I	No	I	EL	L2CTL[L2DO] = 0
		E	Yes	I	EL	L2CTL[L2DO] = 0
		EL	Yes	I	Same	L2CTL[L2DO] = 0
		T	No	I	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus

Table 7-22. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cache-inhibited instruction fetch	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cacheable load (4-state) Cacheable **lwarx** (4-state) dcbt_L1 (4-state) dcbtls_L1 (4-state)	dL1 I	I/T	No	E	Same	L2CTL[L2IO] = 1
		E	Yes	E	I	L2CTL[L2IO] = 1
		EL	Yes	E	T	L2CTL[L2IO] = 1
		I	No	E	E	L2CTL[L2IO] = 0
		E/EL	Yes	E	Same	L2CTL[L2IO] = 0
		T	No	EL	EL	L2CTL[L2IO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited load	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cache-inhibited **lwarx**	N/A	N/A	No	N/A	N/A	No L2 effect
Writeback Store	dL1 I	I/T	No	M	Same	L2 allocates when a line is cast out of L1.
		E	Yes	M	I	
		EL	Yes	M	T	
Writeback **stwcx.**	dL1 I	I/T	No	M	Same	
		E	Yes	M	I	
		EL	Yes	M	T	
Cacheable load (3-state) Cacheable **lwarx** (3-state) dcbt_L1 (3-state) dcbtls_L1 (3-state)	dL1 I	I	No	E/I	I	L2CTL[L2IO] = 1
		T	No	E/I	T	L2CTL[L2IO] = 1
		E	Yes	E/I	I	L2CTL[L2IO] = 1
		EL	Yes	E/I	T	L2CTL[L2IO] = 1
		I	No	E/I	E	L2CTL[L2IO] = 0
dcbt_L2 dcbtst_L2	dL1 I,E	E/EL	Yes	E/I	Same	L2CTL[L2IO] = 0
		T	No	E/I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
dcbtst_L1 dcbtstls_L1	dL1 I	I/T	No	E	Same	
		E	Yes	E	I	
		EL	Yes	E	T	
dcbtls_L2 dcbtstls_L2	dL1 I,E	I	No	I	I	L2CTL[L2IO] = 1
		T	No	I	T	L2CTL[L2IO] = 1
		E	Yes	I	I	L2CTL[L2IO] = 1
		EL	Yes	I	T	L2CTL[L2IO] = 1
		I	No	I	EL	L2CTL[L2IO] = 0
		E/EL	Yes	I	EL	L2CTL[L2IO] = 0
		T	No	I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
Write-through store	dL1 I,E,M	I/T	No	Same	I	
		E/EL	Yes	Same	Same	Read-modify-write

Table 7-22. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cache-inhibited store	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
Cache-inhibited **stwcx.**	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
dcblc_L2 icblc_L2	dL1 I,E,M	I/E	No	Same	Same	
		EL	No	Same	E	
		T	No	Same	I	
Victim castout dcbt_L2 icbt_L2 dcbtst_L2	dL1 M	I/T	No	I	Same	L2CTL[L2IO] = 1. If software sharing cache lines between instructions and data wishes to capture instruction lines in L2 with L2CTL[L2IO] = 1, it must perform **dcbst** to flush the line out of the dL1 before fetching it into L2.
		I	No	I	E	L2CTL[L2IO] = 0
		E/EL	No	I	I/T	L2CTL[L2IO] = 1
			Yes	I	Same	L2CTL[L2IO] = 0
		T	Yes	I	EL	L2CTL[L2IO = 0
dcbtls_L2 icbtls_L2 dcbtstls_L2	dL1 M	I	No	I	EL	An icbtls_L2 that hits modified in L1 cannot be distinguished from dcbtls_L2 and sets the L2 dlock bit. If software shares cache lines between instructions and data and wishes to set ilocks in L2, it must perform **dcbst** to flush the line out of the dL1 before locking it in L2.
		E/EL/T	Yes	I	EL	
Snoop push	dL1 M	I/E	No	I/E	I	
		EL/T	No	I/E	T	Invalidate data, keep lock
dcbf **dcbst**	dL1 M	I/E/EL	No	I	I	
dcbz **dcba**	dL1 I	I/E	No	M	I	
		EL	No	M	T	
dcbi	dL1 I,E,M	I/ E/EL/T	No	I	I	
dcbf **dcbst**	dL1 I,E	I/ E/EL/T	No	I	I	
icbi	iL1 I,V	I/ E/EL/T	No	I	I	

Table 7-23 lists L2 cache state transitions for all system-initiated (non-core) transactions that change the L2. The transaction types and attributes listed follow MPX bus nomenclature, with the addition of write allocate (burst write with L2 cache allocation). Table 7-23 accounts for changes caused by L1 snoop pushes triggered by snoops, listed in Table 7-22.

Table 7-23. State Transitions Due to System-Initiated Transactions

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
Clean IKill	x	x	0	I/E/EL/T	Same	
Flush	x	x	0	I/E/EL/T	I	
Write allocate	0	1	0	I/E/EL/T	EL	Allocate and lock regardless of cache external write (CEW) window
	1	1	0	I/E	E	Allocate regardless of CEW window
				EL/T	EL	
	x	0	0	I/E	I	No allocate if cache-inhibited
				EL/T	T	Invalidate data, keep lock
WWK 32-byte WWF 32-byte WWF atomic	x	1	0	I/E/EL/T	I	Miss in cache external write windows
				I	E/EL	Hit in cache external write window. Set lock if CEW lock attribute set
				EL	Same	
				E	E/EL	
				T	EL	
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock
< 32-byte WWF < 32-byte WWF atomic	x	1	0	I/E	I	Miss in cache external write windows
				EL/T	T	Miss in cache external write windows
				I/T	Same	Hit in CEW window but need burst data
				EL	Same	Hit in cache external write window
				E	E/EL	Hit in cache external write window. Set lock if CEW lock attribute set
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock
Read Read atomic	1	1	0	I/T	Same	
				E	E	
				EL	EL	
	x	0	0	N/A	N/A	No L1/L2 effect
RWNITC	1	1	0	I/L/T	Same	
				E	E	
				EL	EL	
	0	1	0	I	Same	Read-and-clear-lock
				EL	E	
				T	I	
	x	0	0	N/A	N/A	No L1/L2 effect

Table 7-23. State Transitions Due to System-Initiated Transactions (continued)

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
Kill RWITM RWITM atomic RClaim	x	1	0	I/E	I	
				EL/T	T	Invalidate data, keep lock
	x	0	0	I/E/EL/T	Same	

7.10 Initialization/Application Information

This section describes some required steps for initializing the L2 SRAM both in L2 cache mode, and in memory-mapped SRAM mode. Also, it includes some guidelines for error management.

7.10.1 Initialization

This section describes L1 cache and memory-mapped SRAM initialization.

7.10.1.1 L2 Cache Initialization

After power-on reset, the valid bits in the L2 cache status array are in random states. Therefore, it is necessary to perform a Flash invalidate operation before using the array as an L2 cache. This is done by writing a 1 to the L2I bit of the L2 control register (L2CTL). This can be done before or simultaneously with the write that enables the L2 cache. That is, the L2E and L2I bits of L2CTL can be set simultaneously. The L2I bit clears automatically, so no further writes are necessary.

7.10.1.2 Memory-Mapped SRAM Initialization

After power-on reset, the contents of the data and ECC arrays and are random, so all SRAM data must be initialized before it is read. If the cache is initialized by the core or any other device that uses sub-cacheline transactions, ECC error checking should be disabled during the initialization process to avoid false ECC errors generated during the read-modify-write process used for sub-cacheline writes to the SRAM array. This is done by setting the multi- and single-bit ECC error disable bits of the L2 error disable register (L2ERRDIS[MBECCDIS, SBECCDIS]). See Section 7.3.1.5.2, “Error Control and Capture Registers.” If the array is initialized by a DMA engine using cache-line writes, then ECC checking can remain enabled during the initialization process.

7.10.2 Managing Errors

This section describes recommended handling for ECC and tag parity errors.

7.10.2.1 ECC Errors

An individual soft error that causes a single- or multi-bit ECC error can be cleared from the L2 array simply by executing a **dcbf** instruction for the address captured in the L2ERRADDR register. This will invalidate the line in the L2 cache. When the load that caused the ECC error is performed again, the data will be re-allocated into the L2 with ECC bits set properly again.

If the threshold for single bit errors set in the L2ERRCTL register is exceeded, then the L2 cache should be Flash invalidated to clear out all single-bit errors.

Note that no data is lost by executing **dcbf** instructions or Flash invalidate operations because the L2 cache is write-through and contains no modified data.

7.10.2.2 Tag Parity Errors

A tag parity error must be fixed by Flash invalidating the L2 cache. Note that executing a **dcbf** instruction for the address that caused the error to be reported is not sufficient because a tag parity error is seen as an L2 miss and does not cause invalidation of the bad tag. Proper L2 operation cannot be guaranteed if an L2 tag parity error is not repaired by a Flash invalidation of the entire array.

Part III
Memory, Security, and I/O Interfaces

This part defines the memory, security, and I/O interfaces of the MPC8555E and it describes how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- Chapter 8, "e500 Coherency Module," defines the e500 coherency module and how it facilitates communication between the e500 core complex, the L2 cache and the other blocks that comprise the coherent memory domain of the MPC8555E.

  The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8555E.
- Chapter 9, "DDR Memory Controller," describes the DDR SDRAM memory controller of the MPC8555E. 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. A large set of special features like DLL software override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, "Programmable Interrupt Controller," describes the embedded programmable interrupt controller (PIC) of the MPC8555E. This controller is an OpenPIC-compliant interrupt controller that provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to the CPU for servicing.
- Chapter 11, "I^2C Interface," describes the inter-IC (IIC or I^2C) bus controller of the MPC8555E. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8555E powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.
- Chapter 12, "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 13, "Local Bus Controller," describes the local bus controller of the MPC8555E. 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, regular DRAM devices, extended data output DRAM devices, and other peripherals.

- Chapter 14, “Three-Speed Ethernet Controllers,” describes the two three-speed Ethernet controllers on the MPC8555E. These controllers provide 10/100/1Gb Ethernet support with a complete set of media-independent interface options including GMII, RGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8555E memory coherency module.
- Chapter 15, “DMA Controller,” describes the four-channel general-purpose DMA controller of the MPC8555E. The DMA controller transfers blocks of data, independent of the e500 core or external hosts. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 16, “PCI Bus Interface,” describes the PCI controller of the MPC8555E.
- Chapter 17, “Security Engine (SEC) 2.0,” describes the security controller of the MPC8555E.

Chapter 8
e500 Coherency Module

8.1 Introduction

The e500 coherency module (ECM) provides a flexible, easily expandable switching structure for routing e500- and I/O-initiated transactions to target modules on the device. Figure 8-1 shows a high-level block diagram of the ECM.

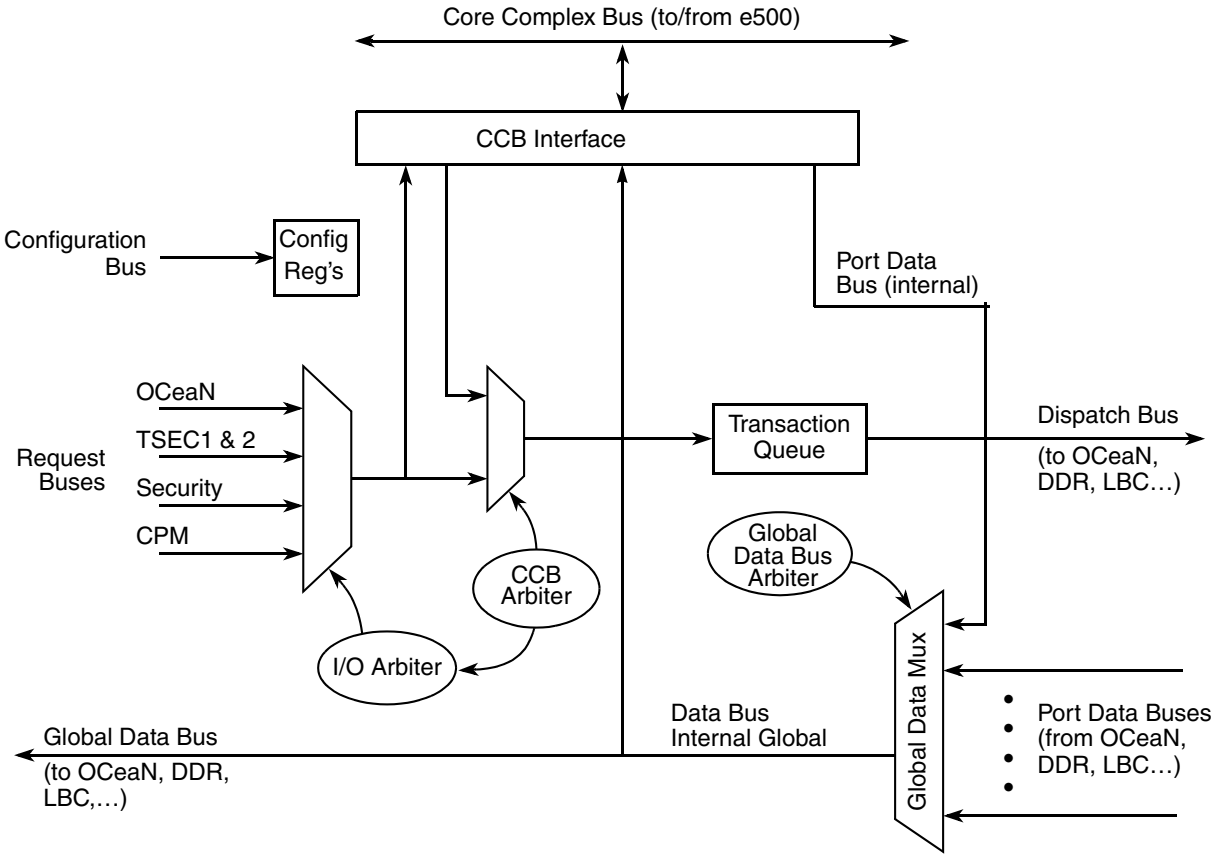


Figure 8-1. e500 Coherency Module Block Diagram

8.1.1 Overview

The ECM routes transactions initiated by the e500 core to the appropriate target interface on the device. In a manner analogous to a bridging router in a local area network, the ECM forwards I/O-initiated transactions that are tagged with the global attribute onto the core complex bus (CCB). This allows on-chip

caches to snoop these transactions as if they were locally initiated and to take actions to maintain coherency across cacheable memory.

8.1.2 Features

The ECM includes these distinctive features:

- Support for e500 core and an L2/SRAM on the CCB, including a CCB arbiter. It sources a 64-bit data bus for returning read data from the ECM to the e500 core and routing write data from the ECM to the L2/SRAM. It sinks a 128-bit data bus for receiving data from the L2/SRAM and a 128-bit write data bus from the e500 core.
- Four connection points for I/O-initiating (mastering into the device) interfaces. One of those connection points services OCeaN initiators, one services security, one services TSEC1 & 2, and one services the CPM and other I/O initiators. The ECM supports five connection points for I/O targets. The DDR memory controller, local bus, OCeaN targets, programmable interrupt controller (PIC), and configuration register access block all have a target port connection to the ECM.
- Split transaction support—separate address and data tenures allow for pipelining of transactions and out-of-order data tenures between initiators and targets.
- Proper ordering of I/O-initiated transactions.
- Speculative read bus for low-latency dispatch of reads to the DDR controller.
- Low-latency path for returning read data from DDR to the e500.
- Error registers trap transactions with invalid addresses. Errors can be programmed to generate interrupts to the e500 core, as described in the following sections:
  - — Section 8.2.1.3, “ECM Error Detect Register (EEDR)”
  - — Section 8.2.1.4, “ECM Error Enable Register (EEER)”
  - — Section 8.2.1.5, “ECM Error Attributes Capture Register (EEATR)”
  - — Section 8.2.1.6, “ECM Error Address Capture Register (EEADR)”
- Errors from reading I/O devices (for example, a master-aborted read transaction on the PCI interface) terminate with data sent to the master with a corrupt attribute. If the master is the e500 core, the ECM asserts *core_fault_in* to the core, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and one of these errors occurs, appropriate interrupts must be enabled to ensure that an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

8.2 Memory Map/Register Definition

Table 8-1 shows the ECM’s memory map. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 8-1. ECM Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*00_0000	8.2.1.2/8-4

Table 8-1. ECM Memory Map (continued)

Local Memory Offset	Register	Access	Reset	Section/Page
0x0_1E00	EEDR—ECM error detect register	Special	0x0000_0000	8.2.1.3/8-4
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.4/8-5
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.5/8-6
0x0_1E10	EEADR—ECM error address capture register	R	0x0000_0000	8.2.1.6/8-7

8.2.1 Register Descriptions

This section consists of detailed descriptions of those registers summarized in Table 8-1. Note that these registers are shown in big-endian format.

8.2.1.1 ECM CCB Address Configuration Register (EEBACR)

The ECM CCB address configuration register, shown in Figure 8-2, controls arbitration and streaming policies for the CCB.

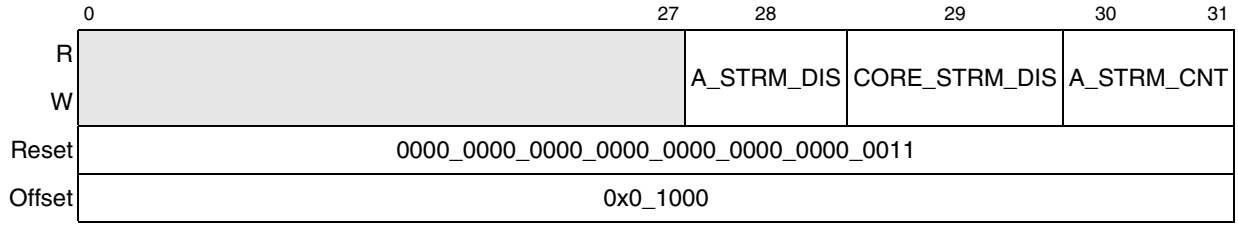


Figure 8-2. ECM CCB Address Configuration Register (EEBACR)

Table 8-2 describes the EEBACR fields.

Table 8-2. EEBACR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	A_STRM_DIS	Controls whether the ECM allows any streaming to occur. 0 Streaming is enabled. 1 Streaming is disabled.
29	CORE_STRM_DIS	With A_STRM_DIS, controls whether the e500 core can stream commands onto the CCB. A_STRM_DIS and CORE_STRM_DIS must both be cleared for the e500 core to be enabled to stream address tenures that it masters. 0 Stream address tenures initiated by the e500 core, provided A_STRM_DIS is cleared. 1 Streaming of address tenures initiated by the e500 core not allowed.
30–31	A_STRM_CNT	Stream count. Specifies the maximum number of transactions that any master can stream (issue sequentially without preemption) on the CCB following an initial transaction. 00 Reserved 01 One transaction can be streamed with the initial transaction. 10 Two transactions can be streamed with the initial transaction. 11 Three transactions can be streamed with the initial transaction (default).

8.2.1.2 ECM CCB Port Configuration Register (EEBPCR)

The ECM CCB port configuration register (EEBPCR) is shown in Figure 8-3.

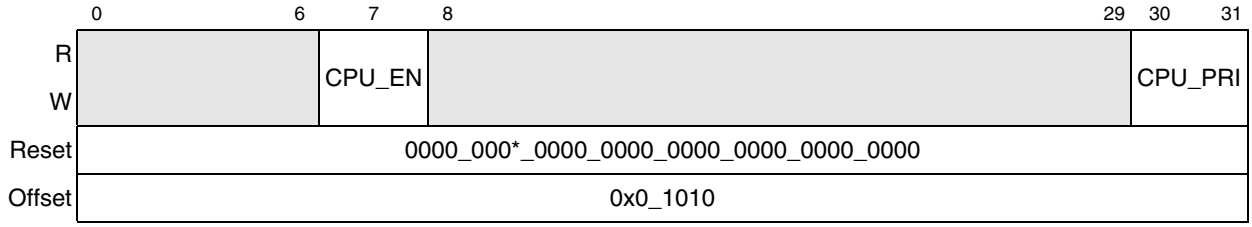


Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)

Table 8-3 describes EEBPCR fields.

Table 8-3. EEBPCR Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	CPU_EN	CPU port enable. Controls boot holdoff mode when the device is an agent of an external host. Specifies whether the e500 core (CPU) port is enabled to run transactions on the CCB. The CPU boot configuration power-on reset pin (cfg_cpu_boot) determines the initial value of this bit. If the pin is sampled as a logic 1 at the negation of reset, the CPU is enabled to boot at the end of the POR sequence. Otherwise, the CPU cannot fetch its boot vector until an external host sets the CPU_EN bit. 0 Boot holdoff mode. CPU arbitration is disabled on the CCB and no bus grants are issued. 1 CPU is enabled and receives bus grants in response to bus requests for the boot vector. After this bit is set, it should not be cleared by software. It is not intended to dynamically enable and disable CPU operation. It is only intended to end boot holdoff mode. See Section 4.4.3.5, “CPU Boot Configuration,” for more information.
8–29	—	Reserved
30–31	CPU_PRI	Specifies the priority level of the e500 core (CPU) port. This priority level is used to determine whether a particular port’s bus request can cause the CCB arbiter to terminate another port’s streaming of address tenures. 00 Lowest priority level 01 Second-lowest priority level 10 Highest priority level 11 Reserved

8.2.1.3 ECM Error Detect Register (EEDR)

The ECM error detect register (EEDR) is shown in Figure 8-4.

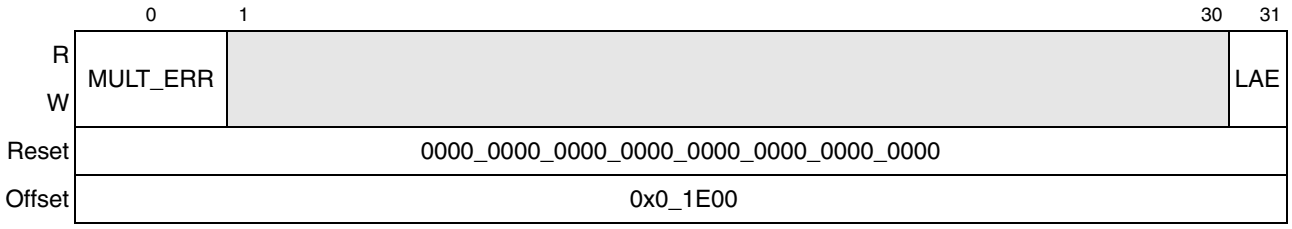


Figure 8-4. ECM Error Detect Register (EEDR)

Table 8-4 describes EEDR fields.

Table 8-4. EEDR Field Descriptions

Bits	Name	Description
0	MULT_ERR	Multiple error. Indicates the occurrence of multiple errors of the same type. Write 1 to clear. 0 Multiple errors of the same type were not detected. 1 Multiple errors of the same type were detected.
1–30	—	Reserved
31	LAE	Local access error. Write 1 to clear. Two cases can generate LAEs: • Transaction does not map to any target. In this case the ECM injects read responses (with the corrupt attribute set) and write data is dropped. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, EEER[LAEE] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." • Source and target IDs indicate that an OCN port initiated a transaction that targets an OCN port. This loopback behavior can result from programming errors where inbound ATMU window targets are inconsistent with targets configured in the local access windows for a given address range. For this type of LAE, the dispatch (to OCN target in this case) is not screened off; the LAE error is reported, but the transaction is still sent to its OCN target. 0 Local access error has not occurred. 1 Local access error occurred.

8.2.1.4 ECM Error Enable Register (EEER)

The ECM error enable register (EEER) shown in Figure 8-5 enables the reporting of error conditions to the e500 core through the internal *int* interrupt signal.

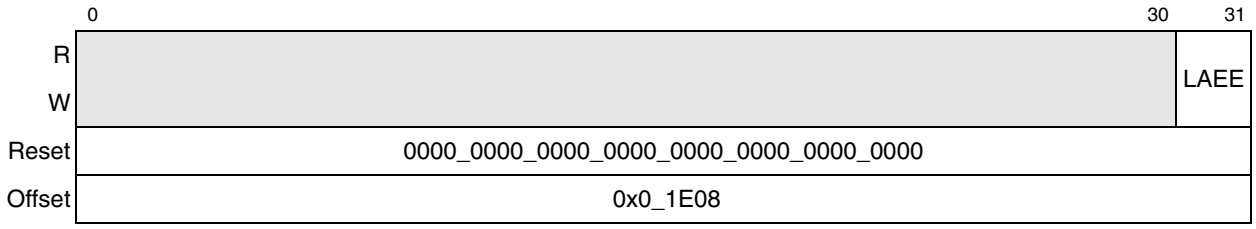


Figure 8-5. ECM Error Enable Register (EEER)

Table 8-5 describes EEER fields.

Table 8-5. EEER Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	LAEE	Local access error enable. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, LAEE must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Disable reporting local access errors as interrupts. 1 Enable reporting local access errors as interrupts.

8.2.1.5 ECM Error Attributes Capture Register (EEATR)

The ECM error attributes capture register (EEATR) is shown in Figure 8-6.

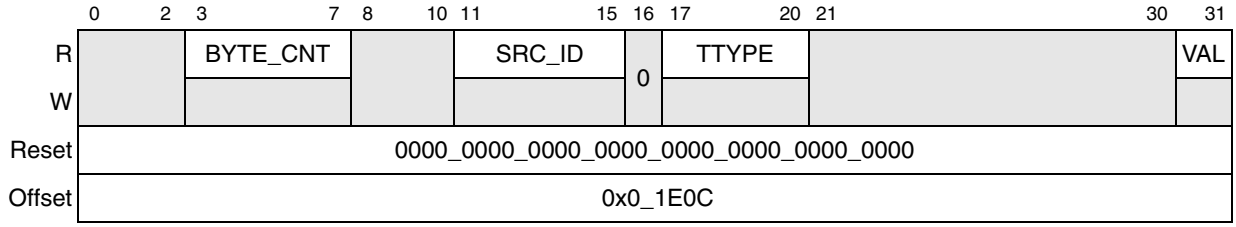


Figure 8-6. ECM Error Attributes Capture Register (EEATR)

Table 8-6 describes EEATR fields.

Table 8-6. EEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	BYTE_CNT	Byte count. Specifies the transaction byte count. 00000 32 bytes 00001 1 byte 00010 2 bytes 00100 4 bytes 01000 8 bytes 10000 16 bytes
8–10	—	Reserved
11–15	SRC_ID	Source ID. Specifies the source device mastering the transaction. 00000 PCI1 interface 00001 PCI2 interface 00010–00110 Reserved 00111 Security 01000–01001 Reserved 01010 Boot sequencer 01011–01111 Reserved 10000 Processor (instruction) 10001 Processor (data) 10010 Reserved 10011 Reserved 10100 CPM 10101 DMA 10110 Reserved 10111 SAP 11000 TSEC1 11001 TSEC2 11010–11111 Reserved
16	—	Reserved
17–20	TTYPE	Transaction type. Defined as follows: 0000 Write 0001 Reserved 0010 Write with allocate 0011 Write with allocate with lock 0100 Address only transaction 0101–0111 Reserved 1000 Read 1001 Read with unlock 101x Reserved 1100 Read with clear atomic 1101 Read with set atomic 1110 Read with decrement atomic 1111 Read with increment atomic

Table 8-6. EEATR Field Descriptions (continued)

Bits	Name	Description
21–30	—	Reserved
31	VAL	Register data valid. 0 ECM error attribute capture register does not contain valid information. 1 ECM error attribute capture register contains valid information.

8.2.1.6 ECM Error Address Capture Register (EEADR)

The ECM error address capture register (EEADR) is shown in Figure 8-7.

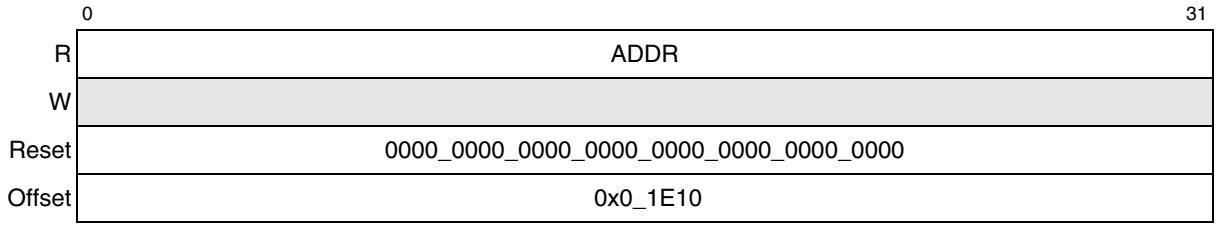


Figure 8-7. ECM Error Address Capture Register (EEADR)

Table 8-7 describes EEADR fields.

Table 8-7. EEADR Field Descriptions

Bits	Name	Description
0–31	ADDR	Address. Specifies the 32-bit address of the transaction. Qualified by EEATR[VAL].

8.3 Functional Description

The following is a very general discussion of ECM operation.

8.3.1 I/O Arbiter

Figure 8-1 shows the I/O arbiter block that manages I/O-initiated address tenure requests arriving on the request buses. Four request buses compete for access to the ECM, which can only process one request at a time. The ECM uses two factors to select the winning request bus: the primary factor is request priority and the secondary factor is longest waiting/least recently granted status. By default all requesters use the lowest priority (priority 0) for requests except for the CPM, which always requests at the highest priority (priority 3). The TSEC controllers dynamically raise and lower their priority levels based on FIFO depth. A starvation avoidance algorithm prevents high-priority requests from indefinitely starving low-priority requesters. The transaction from the winning request bus competes with e500 core requests for the CCB and entry into the transaction queue.

8.3.2 CCB Arbiter

Figure 8-1 shows the CCB arbiter block coordinating the entry of new transactions into the ECM's transaction queue. It handles arbitration for requests to use the CCB from the e500 core and the winning request bus and consequently controls when these new transactions can enter the transaction queue.

Because the CCB operates most efficiently when it streams commands from one initiator, the CCB arbiter alternates grants between streams of transactions from the processor and from the winner of the I/O arbiter. The length of a stream (number of back-to-back transactions) is limited by the A_STRM_CNT field in the EEBACR register. However, the arbiter also uses the priority of the requests to limit streaming. If the priority of a new request is higher than that of a stream in progress, then the higher-priority transaction will interrupt the other stream. The priority of e500 transactions is set by the CPU_PRI field in EEBPCR register.

8.3.3 Transaction Queue

The ECM's transaction queue performs three basic functions: target mapping and dispatching, enforcement of ordering, and enforcement of coherency. The address of each transaction is compared against each local access window, and the transaction is then routed to the appropriate target interface associated with the local access window that the address hits within. Even though the CCB and ECM allow the pipelining of transactions, the address tenures of all transactions issued from masters other than the e500 core (all I/O masters) are strictly ordered and are dispatched to their target interfaces in the same order they are submitted. For those transactions accessing address space marked as snoopable, or space that may be cached by the e500 core, the ECM enforces coherency, snooping those transactions on the CCB, and taking castouts from the e500 core as is necessary.

8.3.4 Global Data Multiplexer

Figure 8-1 shows how the global data multiplexer takes data bus connections and multiplexes them onto one 128-bit global data bus. The global data mux allows initiators of write transactions to route data to their targets and read targets to return data to the initiators.

8.3.5 CCB Interface

Figure 8-1 shows the CCB interface for both CCB address and data tenures. This interface formats CCB address tenures for the ECM transaction queue. It also contains the queueing and buffering needed to manage outstanding CCB data tenures. The buffers receive e500 core-initiated write and I/O-initiated read data (that hit in the L2/SRAM module) from the e500 write (128-bit wide) and read (128-bit wide) data buses and route them through the global data mux to the global data bus. The buffers also receive e500 core-initiated read and I/O-initiated write data (that hit in the L2/SRAM module) from the global data bus and forward them onto the CCB data bus (64 bits).

8.4 Initialization/Application Information

If the e500 core is used to initialize the MPC8555E, the CPU boot configuration power-on reset pin should be pulled high to initially set EEBPCR[CPU_EN]. See Chapter 4, "Reset, Clocking, and Initialization," for more information on power-up reset initialization.

If any device other than the e500 core, such as the boot sequencer or PCI, is used to initialize the MPC8555E, the CPU boot configuration power-on reset pin should be pulled low to initially clear EEBPCR[CPU_EN]. This prevents the e500 core from accessing any configuration registers or local memory space during initialization. However, in any such system, one step near the end of the initialization routine must set EEBPCR[CPU_EN] to re-enable the e500 core. Note that for basic functionality, EEBPCR[CPU_EN] is the only field that must be written (provided a device other than the e500 core is used to initialize the device) in the ECM.

EEBPCR[CPU_PRI] specifies the priority level associated with all e500 core–initiated transactions. This value allows users running time-critical applications to adjust the average response latency of transactions initiated by the core compared to those initiated by I/O masters. This priority level affects whether the e500 core requests can interrupt the streaming of address tenures initiated by (the ECM on behalf of) I/O masters. Only transactions with a priority greater than the current CCB transaction can interrupt streaming. The higher the core's priority, the lower the average latency needed for it to obtain bus grants from the ECM, because it can interrupt lower-priority streaming. The default value of zero gives all core-initiated transactions the lowest priority, which prevents the core from interrupting I/O master transaction streams.

EEBACR[A_STRM_CNT] allows users to balance response latency with throughput and should prove useful in tuning systems with multiple time-critical tasks. The default value of 0b11 causes the ECM to attempt to stream as many as four transactions initiated from the same CCB master. Increasing this value increases the maximum number of transactions that may be streamed together from any one CCB master. Raising this value can increase throughput for high-priority transactions, but may increase latency for lower-priority transactions from another CCB master. Note that the e500 core must also have streaming enabled (through HID1[ASTME]) for the CCB to stream.

Chapter 9
DDR Memory Controller

9.1 Introduction

The fully programmable DDR SDRAM controller supports most first-generation JEDEC standard ×8 or ×16 DDR memories available, including buffered and unbuffered DIMMs. However, mixing 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.

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 sub-banks in each SDRAM chip. A sub-bank is specified by the two least significant bits (lsbs) of a bank address.

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.

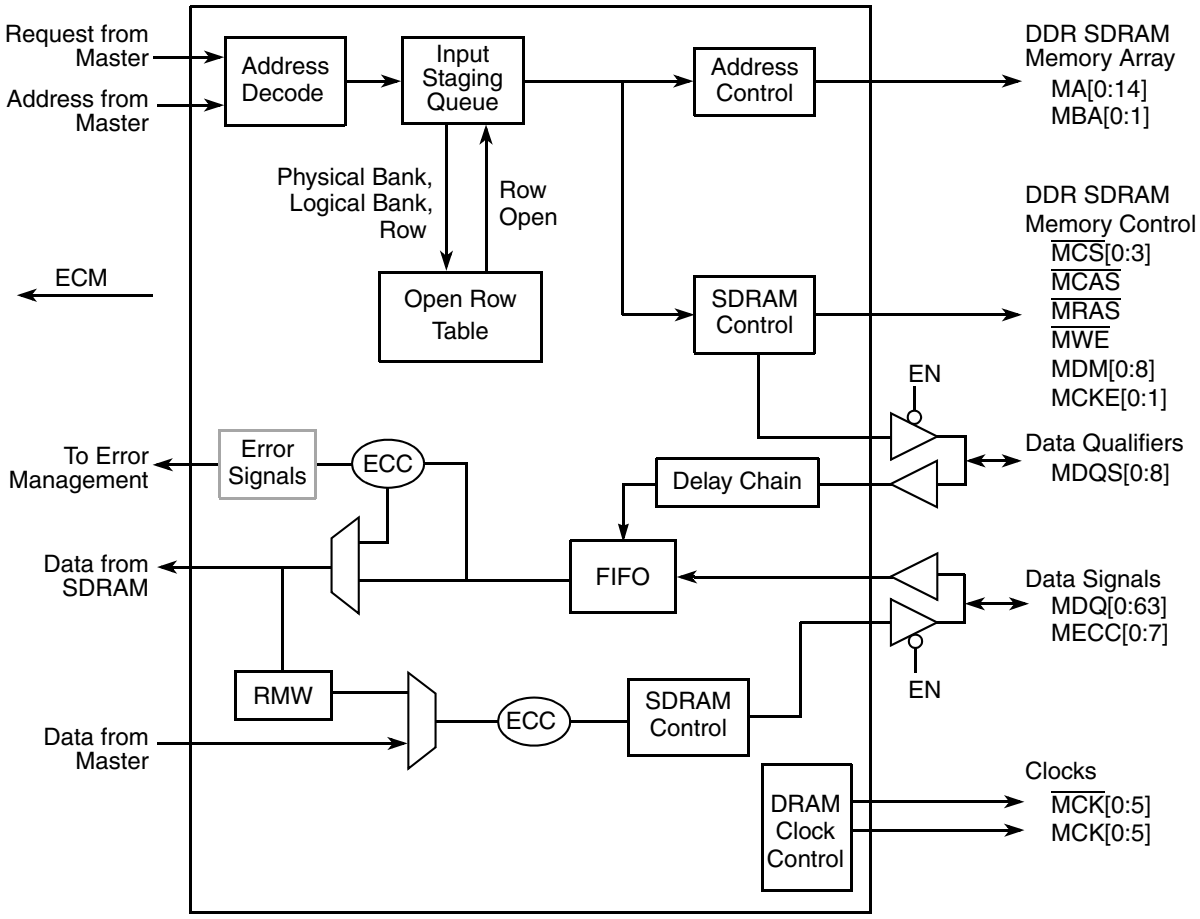


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR SDRAM
- 64-/72-bit SDRAM data bus
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported
  - — As many as four physical banks (chip selects), each bank independently addressable
  - — 64-Mbit to 1-Gbit devices with ×8/×16 data ports (no direct ×4 support)
  - — Unbuffered and registered DIMMs
- Support for data mask signals and read-modify-write for sub-double word writes. Note that 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)
- Two-entry input request queue

- Open page management (dedicated entry for each logical bank)
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

9.2.1 Modes of Operation

The DDR memory controller supports the following modes:

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

9.3 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.

NOTE

A bar over a signal name indicates that the signal is active low, such as $\overline{\text{AS}}$ (address strobe). 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 NMI (nonmaskable interrupt), are referred to as asserted when they are high and negated when they are low.

9.3.1 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 *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* 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	High-Z	1	O

Table 9-1. DDR Memory Interface Signal Summary (continued)

Name	Function/Description	Reset	Pins	I/O
MA[0:14]	Address bus	High-Z	15	O
MBA[0:1]	Logical bank address	High-Z	2	O
$\overline{\text{MCS}}$[0:3]	Chip select	High-Z	4	O
$\overline{\text{MWE}}$	Write enable	High-Z	1	O
$\overline{\text{MRAS}}$	Row address strobe	High-Z	1	O
MDM[0:8]	Data mask	High-Z	9	O
MCK[0:5]	DRAM clock outputs	High-Z	6	O
$\overline{\text{MCK}}$[0:5]	DRAM clock outputs (complement)	High-Z	6	O
MCKE[0:1]	DRAM clock enable	Driven Low	2	O
MDVAL	Memory debug data valid	Zero	1	O
MSRCID[0:4]	Memory debug source ID	All zeros	5	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)	Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
		SDRAM 168-Pin DIMM			SDRAM 168-Pin DIMM
msb	MA14	—		MA5	A5
	MA13	A13		MA4	A4
	MA12	A12		MA3	A3
	MA11	A11		MA2	A2
	MA10	A10(AP)		MA1	A1
	MA9	A9		MA0	A0
	MA8	A8		MBA1	BA1
	MA7	A7	lsb	MBA0	BA0
	MA6	A6			

9.3.2 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(s)	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 DIMM 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 and as 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	Asserted/Negated—Driven high when positive capture data is transmitted/received and driven low when negative capture data is transmitted/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 9-26 for byte lane assignments.
		Timing	Assertion/Negation—If a WRITE command is registered at clock edge *n*, data strobes at the DRAM assert centered in the data “eye” on clock edge *n* + 1. See the JEDEC DDR SDRAM specification for more information.
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is transmitted/received and driven low when negative capture data is transmitted/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 9-26 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(s)	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 20.4.3.2, "Debug Information on ECC Pins," 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[0:14]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[0:14] carry 15 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA[0] 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. SeeTable 9-28 for a complete description of the mapping of these signals.
		Timing	Assertion/Negation—The address is always driven when the memory controller is active. It is valid when a transaction is driven to DRAM (when $\overline{\text{MCS}}n$ is negated). High impedance—When the memory controller is idle.
MBA[0:1]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four 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. MBA[0] 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 9-28 describes the mapping of these signals in all cases.
		Timing	Assertion/Negation—Same timing as MA*n* High-impedance—Same timing as MA*n*
$\overline{\text{MCAS}}$	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.	
		State Meaning	Asserted—Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 9-29 for more information on the states required on $\overline{\text{MCAS}}$ for various other SDRAM commands. Negated—The column address is not guaranteed to be valid.
		Timing	Assertion/Negation—Assertion and negation timing is directed by the values described in Section 9.4.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)." High impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is idle.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal(s)	I/O	Description	
$\overline{\text{MRAS}}$	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.	
		State Meaning	Asserted—Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See Table 9-29 for more information on the states required on $\overline{\text{MRAS}}$ for various other SDRAM commands. Negated—The row address is not guaranteed to be valid.
		Timing	Assertion/Negation—Timing is directed by the values described in Section 9.4.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)." High impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is idle.
$\overline{\text{MCS}}$[0:3]	O	Chip select. Four chip selects supported by the memory controller.	
		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. Negated—Indicates no SDRAM action during the current cycle
		Timing	Assertion/Negation—Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_1. High impedance—Always driven unless the memory controller is disabled
$\overline{\text{MWE}}$	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.	
		State Meaning	Asserted—Indicates a memory write operation. See Table 9-29 for more information on the states required on $\overline{\text{MWE}}$ for various other SDRAM commands. Negated—Indicates a memory read operation
		Timing	Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. Used for write commands. High impedance—$\overline{\text{MWE}}$ is always driven unless the memory controller is idle.
MDM[0:8]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a burst 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); MDM7 corresponds to the LSB. MDM8 corresponds to the ECC byte. Table 9-26 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 MDQx as outputs High impedance—Always driven unless the memory controller is disabled

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(s)	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."	
		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— Source synchronous configuration as defined by the DDR_SDRAM_CLK_CNTL register determines timing relationship.
MCKE[0:1]	O	Clock enable. Two identical output signals (each hereafter referred to simply as MCKE) used as the clock enable to one or more SDRAMs. MCKE can be negated to stop clocking the DDR SDRAM. While this results in system power savings, the user is cautioned to disable SDRAM clocking only when there are no transactions on the interface.	
		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 $\overline{\text{MCK}}$. MCK/$\overline{\text{MCK}}$ are don't cares while MCKE is negated.
		Timing	Assertion/Negation—Similar timing to MA*n* 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 20.4.3, "DDR SDRAM Interface Debug."

9.4 Memory Map/Register Definition

Table 9-5 shows the register memory map for the DDR memory controller. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 9-5. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
0x0_2000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-9
0x0_2008	CS1_BNDS—Chip select 1 memory bounds			
0x0_2010	CS2_BNDS—Chip select 2 memory bounds			
0x0_2018	CS3_BNDS—Chip select 3 memory bounds			
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-10
0x0_2084	CS1_CONFIG—Chip select 1 configuration			
0x0_2088	CS2_CONFIG—Chip select 2 configuration			
0x0_208C	CS3_CONFIG—Chip select 3 configuration			
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.3/9-11

Table 9-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.4/9-12
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.5/9-13
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.6/9-14
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.7/9-15
0x0_2130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0000_0000	9.4.1.8/9-16
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.9/9-17
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.10/9-17
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.11/9-18
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.12/9-19
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.13/9-19
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.14/9-20
0x0_2E40	ERR_DETECT—Memory error detect	Special	0x0000_0000	9.4.1.15/9-20
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.16/9-21
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.17/9-22
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.18/9-22
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.19/9-23
0x0_2E58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.20/9-24

9.4.1 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) shown in Figure 9-2 define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in CS*n*_BNDS should equal the size of physical DRAM. Also, note that EA*n* must be greater than or equal to SA*n*. If the high-order 8 bits of an address are greater than or equal to SA*n*, and they are less than or equal to EA*n*, then chip select *n* will be used.

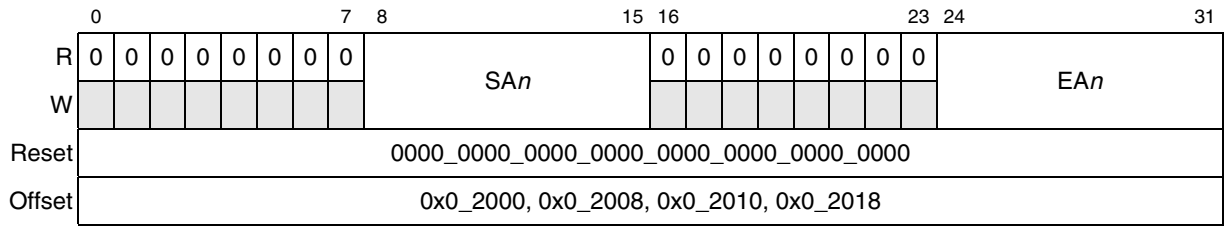


Figure 9-2. Chip Select Bounds Registers (CS*n*_BNDS)

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	SA*n*	Starting address for chip select (bank) *n.* This value is compared against the 8 msbs of the address.
16–23	—	Reserved
24–31	EA*n*	Ending address for chip select (bank) *n.* This value is compared against the 8 msbs of the 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.

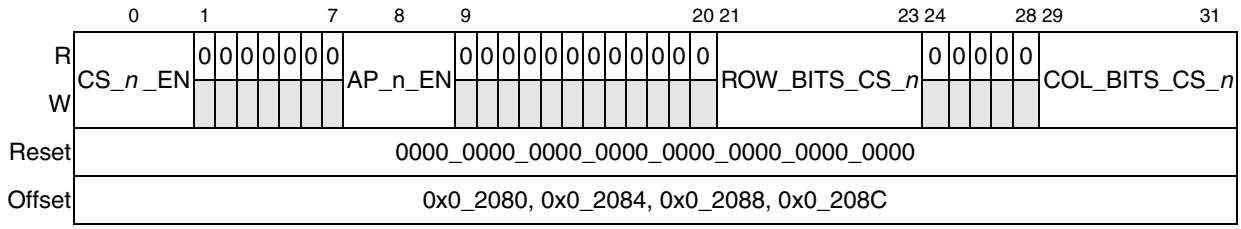


Figure 9-3. Chip Select Configuration Register (CS*n*_CONFIG)

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 *n* enable 0 Chip select *n* is not active. 1 Chip select *n* is active and assumes the state set in CSn_BNDS.
1–7	—	Reserved
8	AP_n_EN	Chip select *n* auto precharge enable 0 Chip select n is auto precharged only if global auto precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select *n* always issues an auto precharge for read and write transactions.
9–20	—	Reserved
21–23	ROW_BITS_CS_*n*	Number of row bits for SDRAM on chip select *n* 000 12 row bits 001 13 row bits 010 14 row bits 011–111 Reserved

Table 9-7. CS*n*_CONFIG Field Descriptions (continued)

Bits	Name	Description
24–28	—	Reserved
29–31	COL_BITS_CS_*n*	Number of column bits for SDRAM on chip select *n* 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 011–111 Reserved

9.4.1.3 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in Figure 9-4, sets the number of clock cycles between various SDRAM control commands.

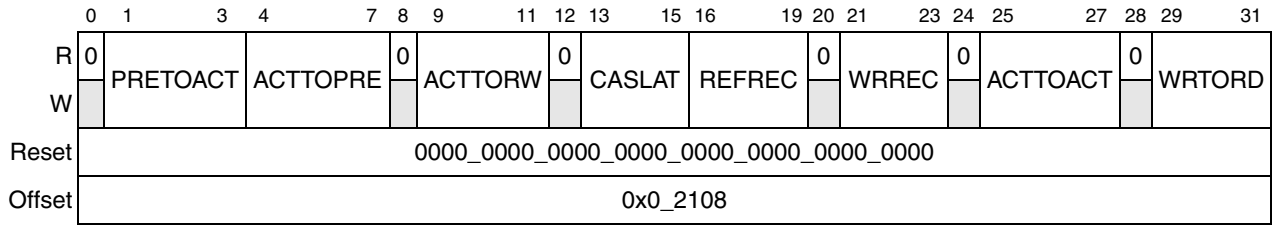


Figure 9-4. DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1)

Table 9-8 describes TIMING_CFG_1 fields.

Table 9-8. 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 Reserved 0001 1 clock 0010 2 clocks 0010 2 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
12	—	Reserved, should be cleared.

Table 9-8. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
13–15	CASLAT	$\overline{\text{MCAS}}$ latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge $n + m$. This value must be programmed at initialization as described in Section 9.4.1.6, "DDR SDRAM Mode Configuration (DDR_SDRAM_MODE).") 000 Reserved 001 1 clock 010 1.5 clocks 011 2 clocks 100 2.5 clocks 101 3 clocks 110 3.5 clocks 111 4 clocks
16–19	REFREC	Refresh recovery time (t_{rfc}). Controls the number of clock cycles from a refresh command until an activate command is allowed. Refresh recovery time is equal to eight plus the REFREC value. 0000 24 clocks 0001 9 clocks 0010 10 clocks 0011 11 clocks ... 1111 23 clocks
20	—	Reserved, should be cleared.
21–23	WRREC	Last data to precharge minimum interval (t_{wr}). Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed. 000 0 clock 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	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). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101–111 Reserved
28	—	Reserved, should be cleared.
29–31	WRTORD	Last write data pair to read command issue interval (t_{wtr}). Number of clock cycles between the last write data pair and the subsequent read command to the same physical bank. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

9.4.1.4 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in Figure 9-5, sets the clock delay to data for writes.

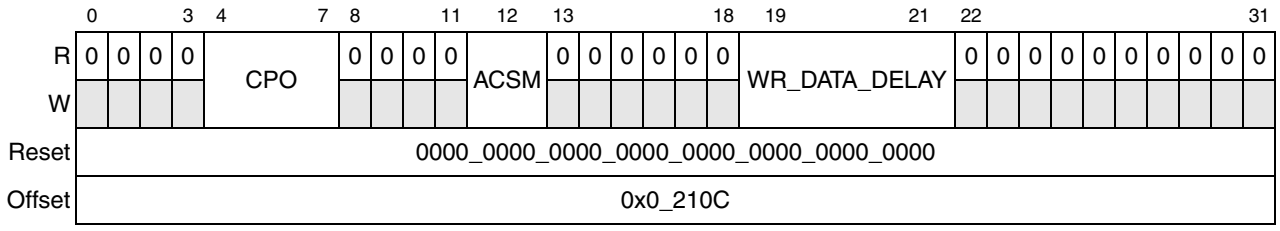


Figure 9-5. DDR SDRAM Timing Configuration Register 2 (TIMING_CFG_2)

Table 9-9 describes the TIMING_CFG_2 fields.

Table 9-9. TIMING_CFG_2 Register Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	CPO	$\overline{\text{MCAS}}$-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. 0000 Default. $\overline{\text{MCAS}}$ to preamble is defined as CASLAT + 1 0001 ⌈CASLAT⌉ 0010 ⌈CASLAT⌉ + 1/2 0011 ⌈CASLAT⌉ + 1 0100 ⌈CASLAT⌉ + 3/2 0101 ⌈CASLAT⌉ + 2 0110 ⌈CASLAT⌉ + 5/2 0111 ⌈CASLAT⌉ + 3 1000 ⌈CASLAT⌉ + 7/2 1001 ⌈CASLAT⌉ + 4 1010 ⌈CASLAT⌉ + 9/2 1011 ⌈CASLAT⌉ + 5 1100–1111 Reserved
8–11	—	Reserved
12	ACSM	Address and control shift mode 0 The DRAM address and control buses are output in the default mode. 1 The DRAM address and control buses are delayed by 1/2 DRAM cycle before being driven onto the pins.
13–18	—	Reserved
19–21	WR_DATA_DELAY	Write command to write data strobe timing adjustment. Controls the amount of delay applied to the data and data strobes for writes. 000 0 clock delay 001 2/8 clock delay (recommended) 010 4/8 clock delay 011 6/8 clock delay 100 1 clock delay 101–111 Reserved
22–31	—	Reserved

9.4.1.5 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in Figure 9-6, enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMS, and dynamic power management.

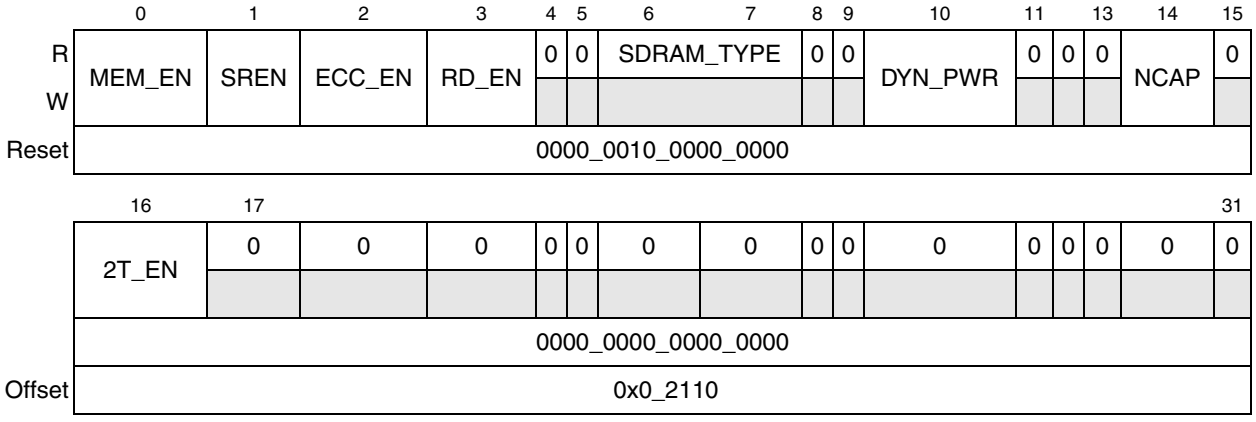


Figure 9-6. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

Table 9-10 describes the DDR_SDRAM_CFG fields.

Table 9-10. 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 or soft-stop. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep or soft-stop. 1 SDRAM self refresh is enabled during sleep or soft-stop
2	ECC_EN	ECC enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 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.
4–5	—	Reserved
6–7	SDRAM_TYPE	Type of SDRAM device to be used 00–01 Reserved 10 DDR SDRAM 11 Reserved
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.
11–13	—	Reserved
14	NCAP	Non-concurrent auto precharge 0 SDRAMs in system support concurrent auto precharge. 1 SDRAMs in system do not support concurrent auto precharge.
15	—	Reserved
16	2T_EN	2T timing enable 0 1T timing is used. The SDRAM command/address are held for only 1 cycle on the SDRAM bus. 1 2T timing is enabled. The SDRAM command/address are held for 2 full cycles on the SDRAM bus for every SDRAM transaction. However, the chip select is only held for the second cycle.
17–31	—	Reserved

9.4.1.6 DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register, shown in Figure 9-7, sets the values loaded into the DDR’s mode registers.

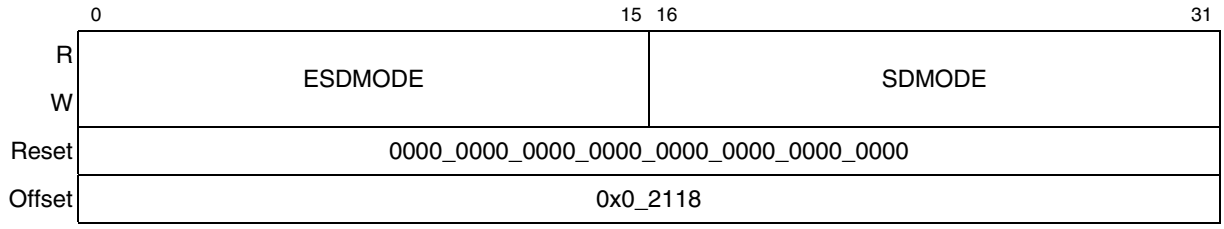


Figure 9-7. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

Table 9-11 describes the DDR_SDRAM_MODE fields.

Table 9-11. DDR_SDRAM_MODE Field Descriptions

Bits	Name	Description
0–15	ESDMODE [0:15]	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. The value of ESDMODE[1:15] is driven onto MA[14:0] during the EXTENDED MODE REGISTER SET operation of the initialization sequence. The lsb of ESDMODE (ESDMODE[15]) is driven onto MA[0] and ESDMODE[1] is driven onto MA[14].
16–31	SDMODE [0:15]	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. The value of SDMODE[1:15] is driven onto MA[14:0] during the MODE REGISTER SET operation of the initialization sequence. The lsb of SDMODE (SDMODE[15]) is driven onto MA[0] and SDMODE[1] is driven onto MA[14]. Because the memory controller forces SDMODE[3–8] to certain values depending upon the state of the initialization sequence (for resetting the SDRAM's DLL) the corresponding bits of this field is ignored by the memory controller.

9.4.1.7 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in Figure 9-8, 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.

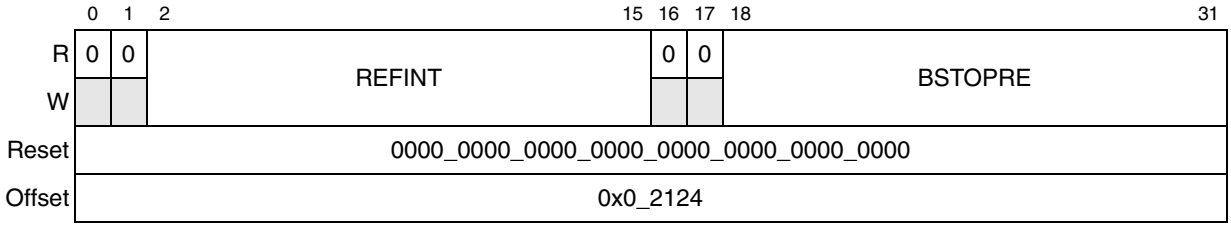


Figure 9-8. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

Table 9-12 describes the DDR_SDRAM_INTERVAL fields.

Table 9-12. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–15	REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. One row is 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. Note that REFINT must be set to a non-zero value in order for the DDR to enter sleep mode. See Section 18.5.1.5.3, “Sleep Mode,” for additional details.
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.

9.4.1.8 DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register, shown in Figure 9-9, provides a source synchronous option, along with a 1/4 cycle clock adjustment.

NOTE

The DDR_SDRAM_CLK_CNTL[SS_EN] default value out of reset is 0 and this must be set to a 1 for proper operation.

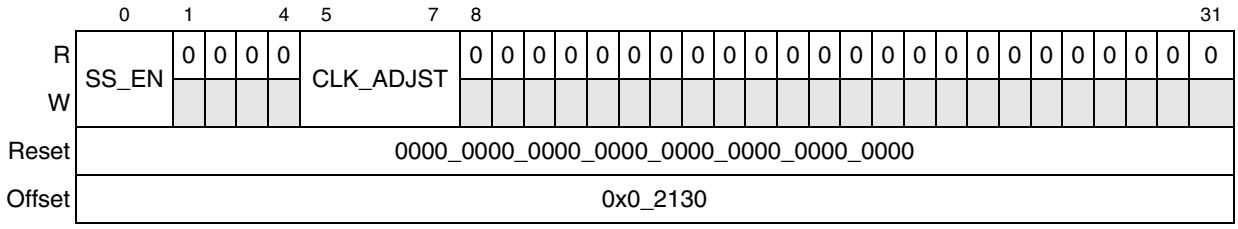


Figure 9-9. DDR SDRAM Clock Control Register (DDR_SDRAM_CLK_CNTL)

Table 9-13 describes the DDR_SDRAM_CLK_CNTL fields.

Table 9-13. DDR_SDRAM_CLK_CNTL Field Descriptions

Bits	Name	Description
0	SS_EN	Source synchronous enable. This bitfield must be set during initialization. See Section 9.6.1, “DDR SDRAM Initialization Sequence,” for details. 0 Reserved 1 The address and command are sent to the DDR SDRAMs source synchronously.
1–4	—	Reserved

Table 9-13. DDR_SDRAM_CLK_CNTL Field Descriptions (continued)

Bits	Name	Description
5–7	CLK_ADJST	Clock adjust 000 Applied clock (MCK/MCK_B) is launched aligned with address/command 001 Applied clock (MCK/MCK_B) is launched 1/4 of one SDRAM clock cycle after address/command 010 Applied clock (MCK/MCK_B) is launched 1/2 of one SDRAM clock cycle after address/command 011 Applied clock (MCK/MCK_B) is launched 3/4 of one SDRAM clock cycle after address/command 100 Applied clock (MCK/MCK_B) is launched 1 SDRAM clock cycle after address/command. Note that this setting causes the applied clock and address/command to be aligned just as a setting of 000 101–111 Reserved
8–31	—	Reserved

9.4.1.9 Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)

The memory data path error injection mask high register is shown in Figure 9-10.

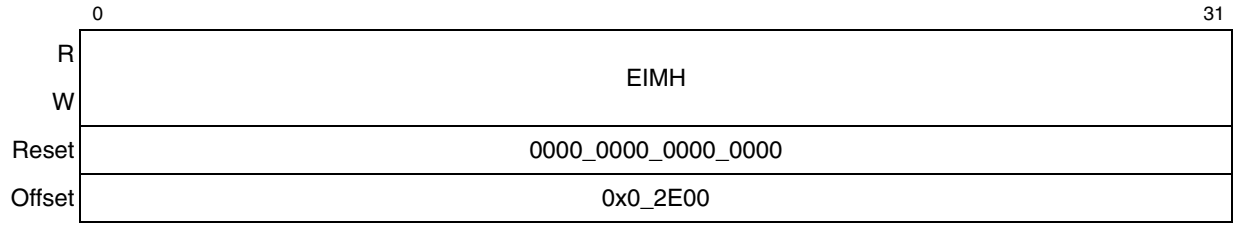


Figure 9-10. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

Table 9-14 describes the DATA_ERR_INJECT_HI fields.

Table 9-14. 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.

9.4.1.10 Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)

The memory data path error injection mask low register is shown in Figure 9-11.

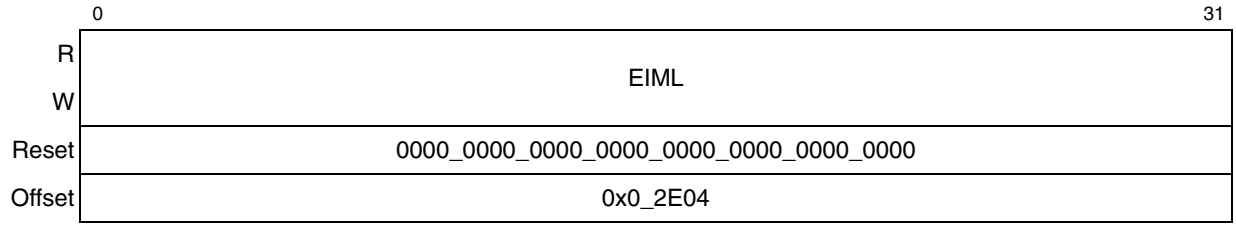


Figure 9-11. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

Table 9-15 describes the DATA_ERR_INJECT_LO fields.

Table 9-15. 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.

9.4.1.11 Memory Data Path Error Injection Mask ECC (ECC_ERR_INJECT)

The memory data path error injection mask ECC register, shown in Figure 9-12, sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte.

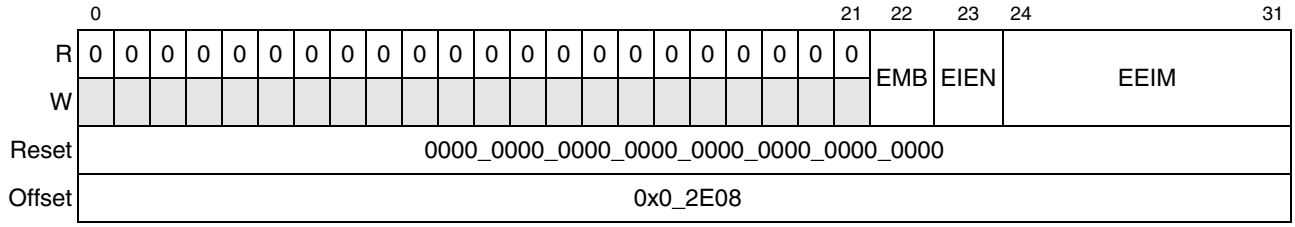


Figure 9-12. Memory Data Path Error Injection Mask ECC Register (ECC_ERR_INJECT)

Table 9-16 describes the ECC_ERR_INJECT fields.

Table 9-16. ECC_ERR_INJECT Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	EMB	ECC mirror byte 0 Mirror byte functionality disabled 1 Mirror the most significant data path byte onto the ECC byte.
23	EIEN	Error injection enable 0 Error injection disabled 1 Error injection enabled. This applies to the data mask bits, the ECC mask bits, and the ECC mirror bit.
24–31	EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

9.4.1.12 Memory Data Path Read Capture High (CAPTURE_DATA_HI)

The memory data path read capture high register, shown in Figure 9-13, stores the high word of the read data path during error capture.

	0	31
R W	ECHD	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x0_2E20	

Figure 9-13. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

Table 9-17 describes the CAPTURE_DATA_HI fields.

Table 9-17. CAPTURE_DATA_HI Field Descriptions

Bits	Name	Description
0–31	ECHD	Error capture high data path. Captures the high word of the data path when errors are detected.

9.4.1.13 Memory Data Path Read Capture Low (CAPTURE_DATA_LO)

The memory data path read capture low register, shown in Figure 9-14, stores the low word of the read data path during error capture.

	0	31
R W	ECLD	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x0_2E24	

Figure 9-14. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

Table 9-18 describes the CAPTURE_DATA_LO fields.

Table 9-18. 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.

9.4.1.14 Memory Data Path Read Capture ECC (CAPTURE_ECC)

The memory data path read capture ECC register, shown in Figure 9-15, stores the ECC syndrome bits that were on the data bus when an error was detected.

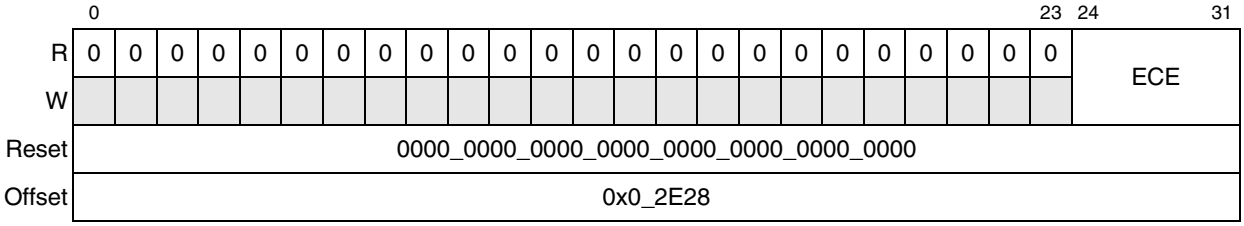


Figure 9-15. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 9-19 describes the CAPTURE_ECC fields.

Table 9-19. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected.

9.4.1.15 Memory Error Detect (ERR_DETECT)

The memory error detect register, shown in Figure 9-16, 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 high value to the bit. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

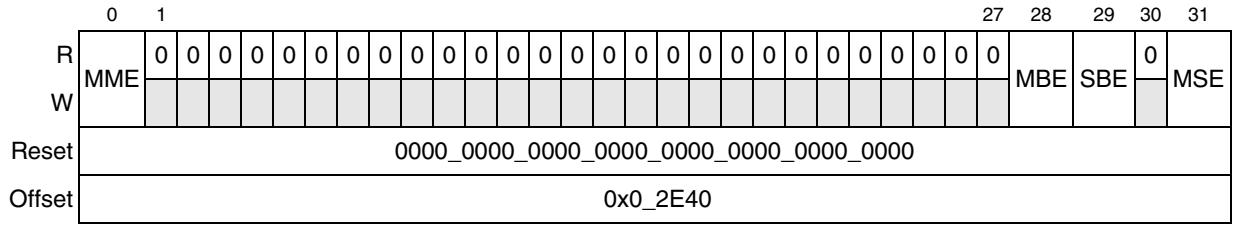


Figure 9-16. Memory Error Detect Register (ERR_DETECT)

Table 9-20 describes the ERR_DETECT fields.

Table 9-20. ERR_DETECT Field Descriptions

Bits	Name	Description
0	MME	Multiple memory errors. This bit is cleared by software writing a 1. 0 Multiple memory errors of the same type were not detected. 1 Multiple memory errors of the same type were detected.
1–27	—	Reserved

Table 9-20. ERR_DETECT Field Descriptions (continued)

Bits	Name	Description
28	MBE	Multiple-bit error. This bit is cleared by software writing a 1. 0 A multiple-bit error has not been detected. 1 A multiple-bit error has been detected.
29	SBE	Single-bit ECC error. This bit is cleared by software writing a 1. 0 The number of single-bit ECC errors detected has not crossed the threshold set in ERR_SBE[SBET]. 1 The number of single-bit ECC errors detected crossed the threshold set in ERR_SBE[SBET].
30	—	Reserved
31	MSE	Memory select error. This bit is cleared by software writing a 1. 0 A memory select error has not been detected. 1 A memory select error has been detected.

9.4.1.16 Memory Error Disable (ERR_DISABLE)

The memory error disable register, shown in Figure 9-17, allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

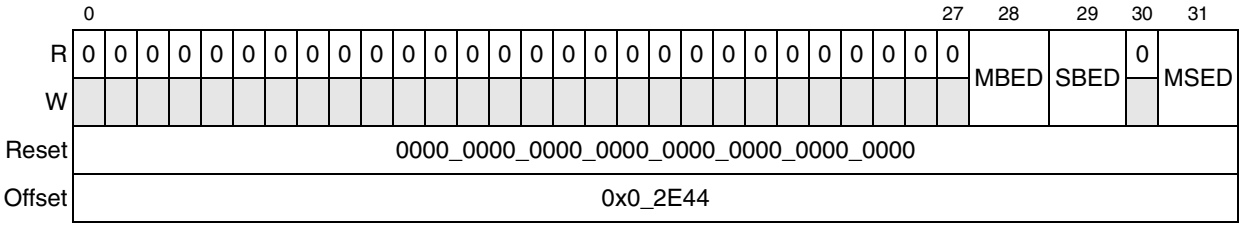


Figure 9-17. Memory Error Disable Register (ERR_DISABLE)

Table 9-21 describes the ERR_DISABLE fields.

Table 9-21. ERR_DISABLE Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	MBED	Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. Note that uncorrectable read errors cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBED must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
29	SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.
30	—	Reserved
31	MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

9.4.1.17 Memory Error Interrupt Enable (ERR_INT_EN)

The memory error interrupt enable register, shown in Figure 9-18, enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal *int* signal is asserted to the programmable interrupt controller (PIC).

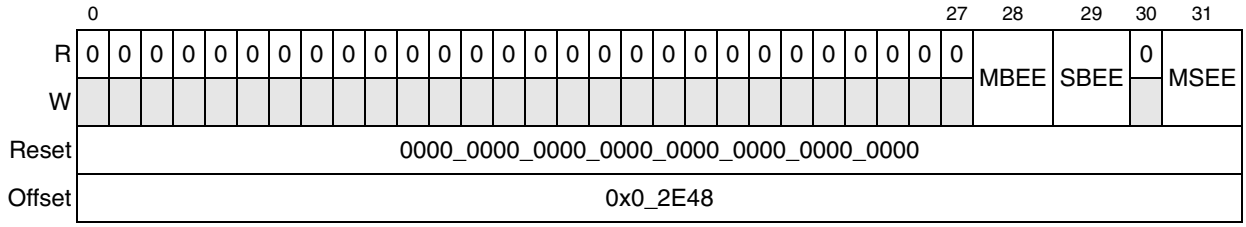


Figure 9-18. Memory Error Interrupt Enable Register (ERR_INT_EN)

Table 9-22 describes the ERR_INT_EN fields.

Table 9-22. ERR_INT_EN Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and MBEE and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
29	SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
30	—	Reserved
31	MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

9.4.1.18 Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)

The memory error attributes capture register, shown in Figure 9-19, sets attributes for errors including type, size, source, and others.

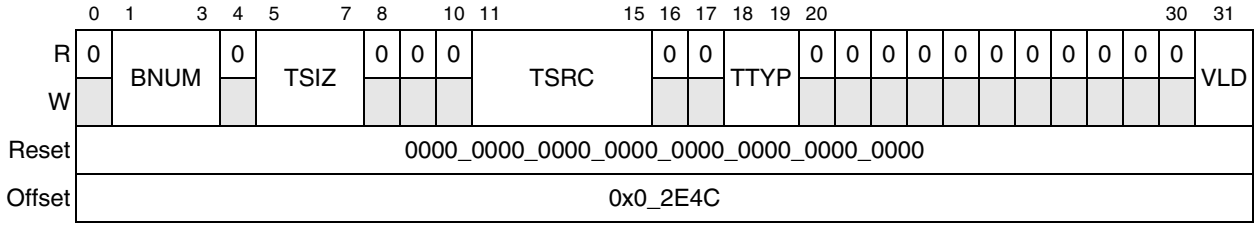


Figure 9-19. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

Table 9-23 describes the CAPTURE_ATTRIBUTES fields.

Table 9-23. CAPTURE_ATTRIBUTES Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number. Captures the data beat number for the detected error. Relevant only for ECC errors.
4	—	Reserved
5–7	TSIZ	Transaction size for the error. Captures the transaction size in double words.
8–10	—	Reserved
11–15	TSRC	Transaction source for the error 00000 PCI 00001–00011 Reserved 00100 Local bus 00101–00111 Reserved 01000 Configuration space 01001 Reserved 01010 Boot sequencer 01011 Reserved 01100 Reserved 01101–01111 Reserved 10000 Processor (instruction) 10001 Processor (data) 10010–10011 Reserved 10100 CPM 10101 DMA 10110 Reserved 10111 SAP 11000 TSEC1 11001 TSEC2 11010 Reserved 11011–11111Reserved
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.

9.4.1.19 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in Figure 9-20, holds the 32 lsbs of a transaction when a DDR ECC error is detected.

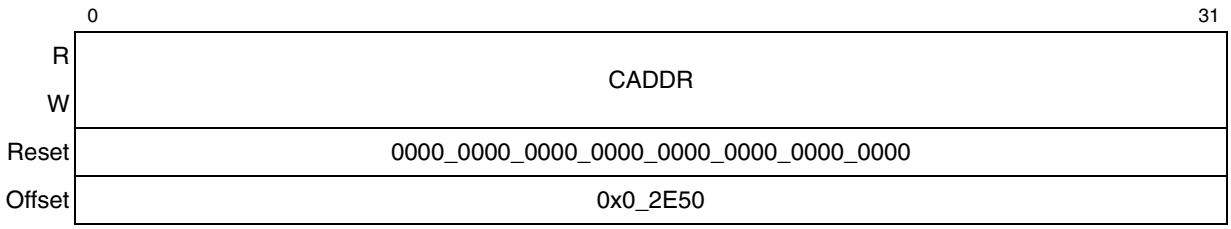


Figure 9-20. Memory Error Address Capture Register (CAPTURE_ADDRESS)

Table 9-24 describes the CAPTURE_ADDRESS fields.

Table 9-24. 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.

9.4.1.20 Single-Bit ECC Memory Error Management (ERR_SBE)

The single-bit ECC memory error management register, shown in Figure 9-21, 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.

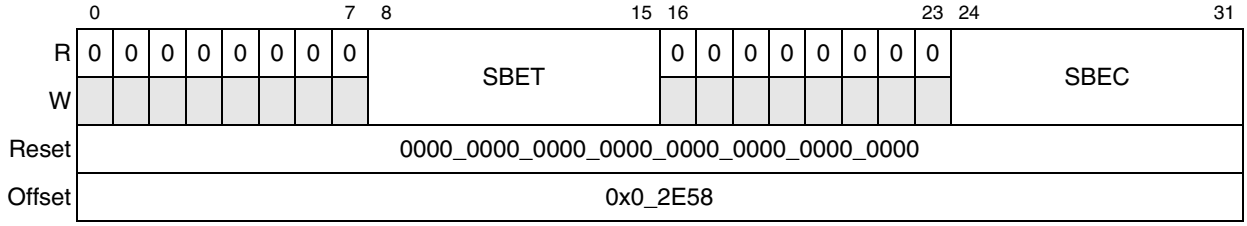


Figure 9-21. Single-Bit ECC Memory Error Management Register (ERR_SBE)

Table 9-25 describes the ERR_SBE fields.

Table 9-25. 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 DDR SDRAMs (first generation dual data rate). The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

Figure 9-22 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 then loaded into the input staging queue with the decoded information. The lower two entries of the input queue are compared with values in the row open table to determine if the address maps to an open page. If the address from either entry does not map to an open

page, an activate command is issued for the entry that did not hit an open page, with the lowest entry having priority over the next lowest. Commands are always issued from the lowest input queue entry.

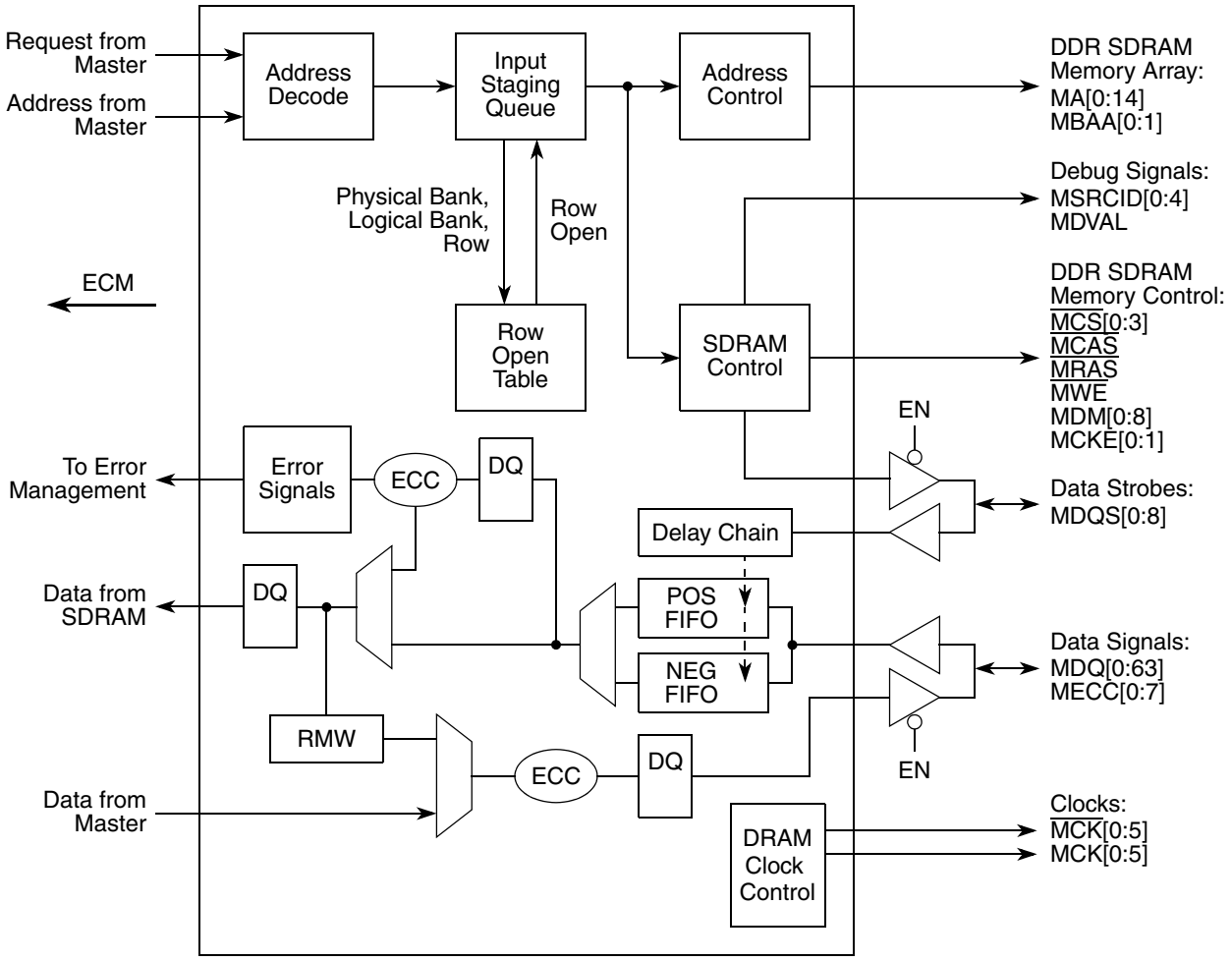


Figure 9-22. DDR Memory Controller Block Diagram

The DDR SDRAM interface supports as many as four physical banks of 64/72 bit-wide memory. A bank size up to 1 Gbyte is supported, providing up to a maximum of 4 Gbytes of DDR main memory. However, 4 Gbytes would span the entire 32-bit address space, and some space must be reserved for boot ROM, configuration registers, and other important addressable locations.

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-bit data bus, detects all double-bit errors within the 64-bit data bus, and detects all errors within a nibble. The controller allows as many as 16 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

Read and write accesses to the DDR SDRAM are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (2, 4, or 8) in a programmed sequence (sequential

or interleaved). 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 is implemented in the controller with delay chains for the data strobe signals during reads and a delay chain on the data multiplexer select during writes.

When ECC is enabled, one clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

Figure 9-23 shows an example DDR SDRAM configuration with four physical banks.

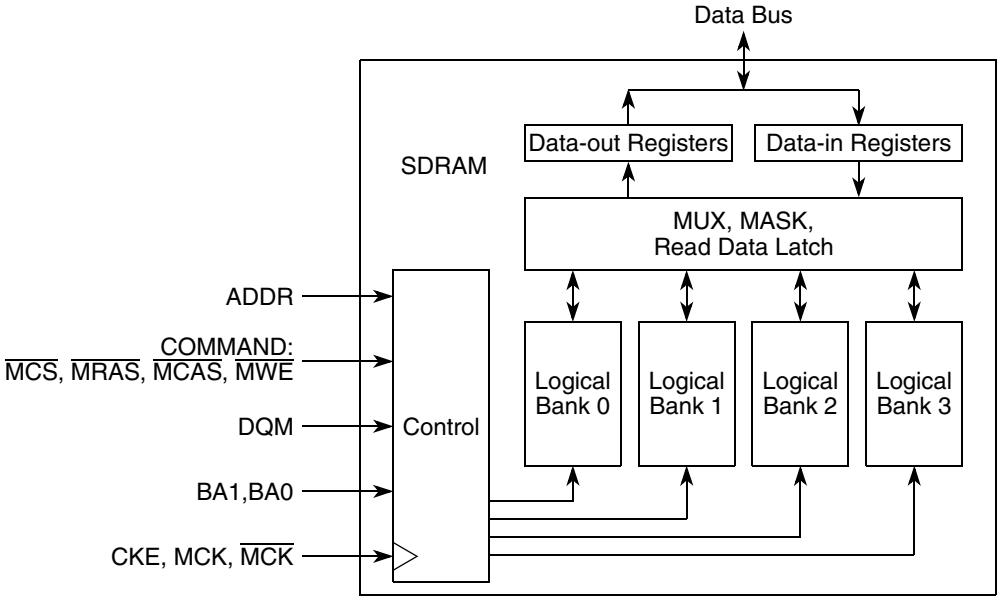


Figure 9-23. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-24 shows some typical signal connections.

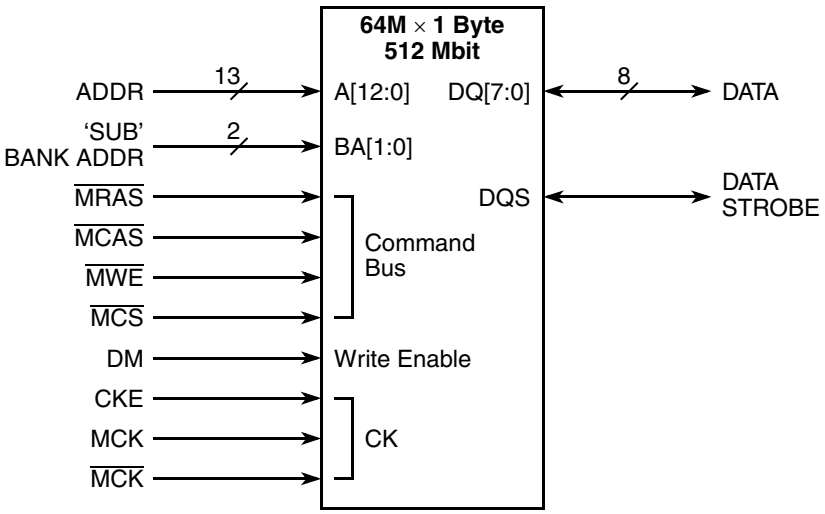
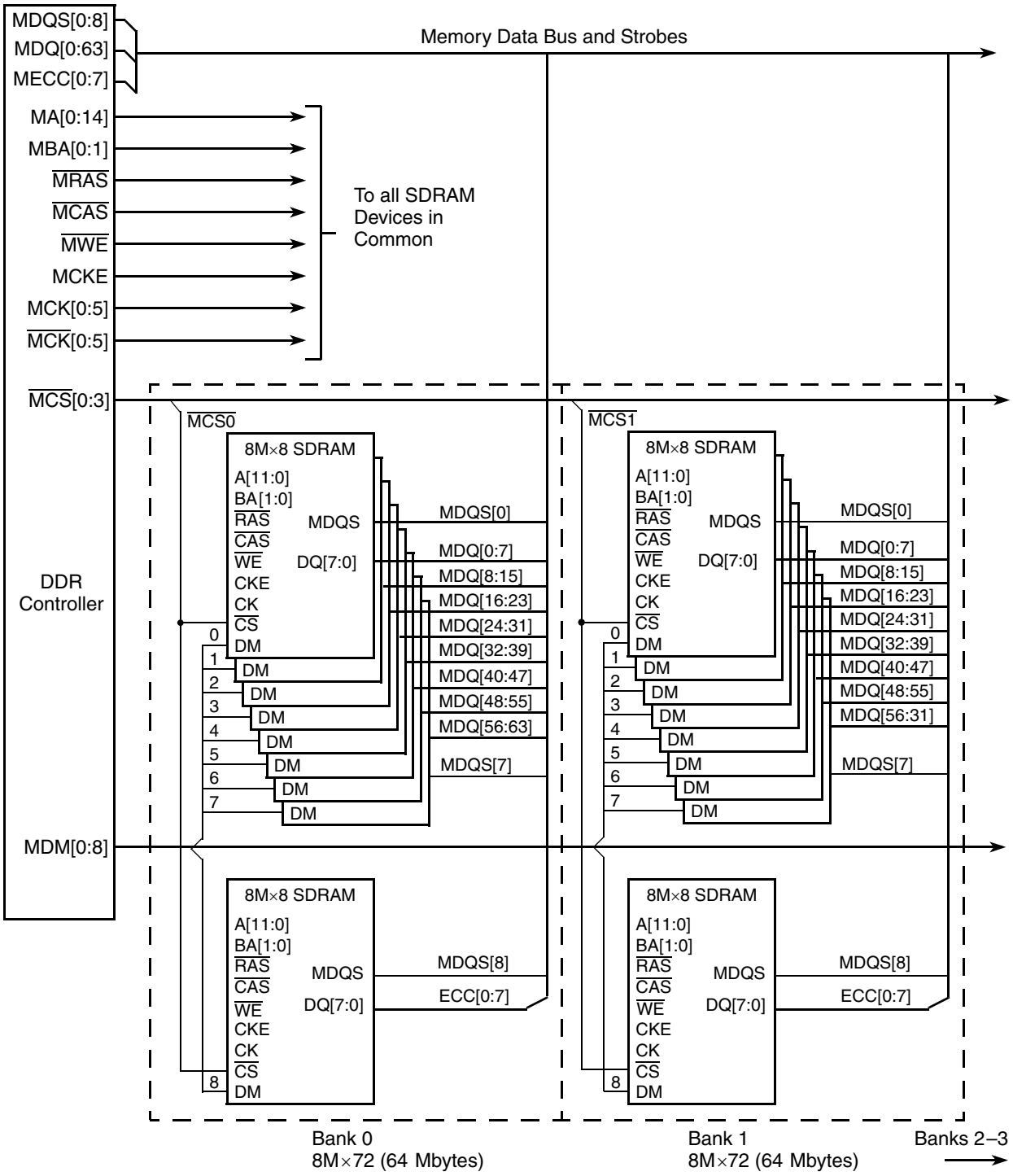


Figure 9-24. Typical DDR SDRAM Interface Signals

Figure 9-25 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M × 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering. Analysis of the MPC8555E 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 MPC8555E 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 MCS[0:3],MCK[0:5], MDM[0:8], and the data bus signals.
2. Each of the MCS[0:3] signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. MCK[0:5] may be apportioned among all memory devices. Complementary bus is not shown.

Figure 9-25. Example 256-Mbyte DDR SDRAM Configuration with ECC

Section 9.5.13, "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. Fourteen multiplexed address signals and two logical bank select signals support device densities from 64 Mbits to 1 Gbit. Four chip select ($\overline{CS}$) signals support up to 2 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 bits wide, 72 bits with ECC. The MPC8555E DDR memory controller supports physical bank sizes from 32 Mbytes to 1 Gbyte. The physical banks can be constructed using ×8, or ×16 memory devices. The memory technologies supported are 64, 128, 256, and 512 Mbits, and 1 Gbit. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

An 8-bit DDR SDRAM device has a DQM signal and 8 data signals (DQ[0:7]). A 16-bit DDR SDRAM device has 2 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 9-26 shows the DDR memory controller's relationships between data byte lane 0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63].

Table 9-26. 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 14 memory address signals and 2 logical bank select signals, individual physical banks may be implemented with memory devices requiring fewer than 28 address bits. Each physical bank may be individually configured to provide from 12 to 14 row address bits, plus 2 logical bank-select bits and from 8–11 column address bits. Table 9-27 describes DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 27 total address bits.

Table 9-27. Supported DDR SDRAM Device Configurations

SDRAM Device	Device Configuration	Row × Column Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits × 8	12 × 9	64 Mbytes	256 Mbytes
64 Mbits	4 Mbits × 16	12 × 8	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits × 8	12 × 10	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits × 16	12 × 9	64 Mbytes	256 Mbytes
256 Mbits	32 Mbits × 8	13 × 10	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits × 16	13 × 9	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits × 8	13 × 11	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits × 16	13 × 10	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits × 8	14 × 11	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits × 16	14 × 10	512 Mbytes	2 Gbytes

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in Section 9.5.13, “Error Management.”

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

9.5.2 DDR SDRAM Address Multiplexing

Table 9-28 shows 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 for reads and writes, so the column address can never use MA[10].

Table 9-28. DDR SDRAM Address Multiplexing

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 × 11	$\overline{MRAS}$			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	
14 × 10	$\overline{MRAS}$				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	
13 × 11	$\overline{MRAS}$				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	
13 × 10	$\overline{MRAS}$					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	
13 × 9	$\overline{MRAS}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{MCAS}$																					8	7	6	5	4	3	2	1	0	
12 × 10	$\overline{MRAS}$						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	
12 × 9	$\overline{MRAS}$							11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{MCAS}$																					8	7	6	5	4	3	2	1	0	
12 × 8	$\overline{MRAS}$								11	10	9	8	7	6	5	4	3	2	1	0											
	MBA																				1	0									
	$\overline{MCAS}$																						7	6	5	4	3	2	1	0	

9.5.3 JEDEC Standard DDR SDRAM Interface Commands

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 9-29) 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 burst size, which is set to four by the DDR memory controller.
- Refresh (similar to $\overline{\text{MCAS}}$ before $\overline{\text{MRAS}}$)—Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh, row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh.
- Mode register set (for configuration)—Allows setting of DDR SDRAM options. These options are: $\overline{\text{MCAS}}$ latency, burst type, and burst length. $\overline{\text{MCAS}}$ latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide $\overline{\text{MCAS}}$ latency {1,2,3}, some provide $\overline{\text{MCAS}}$ latency {1,2,3,4}, and so on). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory controller after DDR_SDRAM_CFG[MEM_EN] is set.
- 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, all logical banks are in a precharged state.

Table 9-29. SDRAM Command Table

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X

Table 9-29. SDRAM Command Table (continued)

Operation	CKE Prev.	CKE Current	$\overline{MCS}$	$\overline{MRAS}$	$\overline{MCAS}$	$\overline{MWE}$	MBA	MA10	MA
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column
Write with auto precharge	H	H	L	H	L	L	Logical bank select	H	Column
Mode register set	H	H	L	L	L	L	Opcode	Opcode	Opcode and mode
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

9.5.4 SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four-beat burst read, but ignores the last three beats. Single-beat writes are performed by masking the last three beats of the four-beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than 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 9-30 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-30. 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.
ACTOPRE	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.
ACTORW	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.
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.

Table 9-30. DDR SDRAM Interface Timing Intervals (continued)

Timing Intervals	Definition
CASLAT	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 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.
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. One row is refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface.
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.
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.
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.

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_1 and TIMING_CFG_2 registers as described in Section 9.4.1.3, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” and Section 9.4.1.4, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)”) 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 9-26 through Figure 9-28 show DDR SDRAM timing for various types of accesses; see Figure 9-26 for a back-to-back burst read operation, Figure 9-27 for a single-beat write operation, and Figure 9-28 for a burst-write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads.

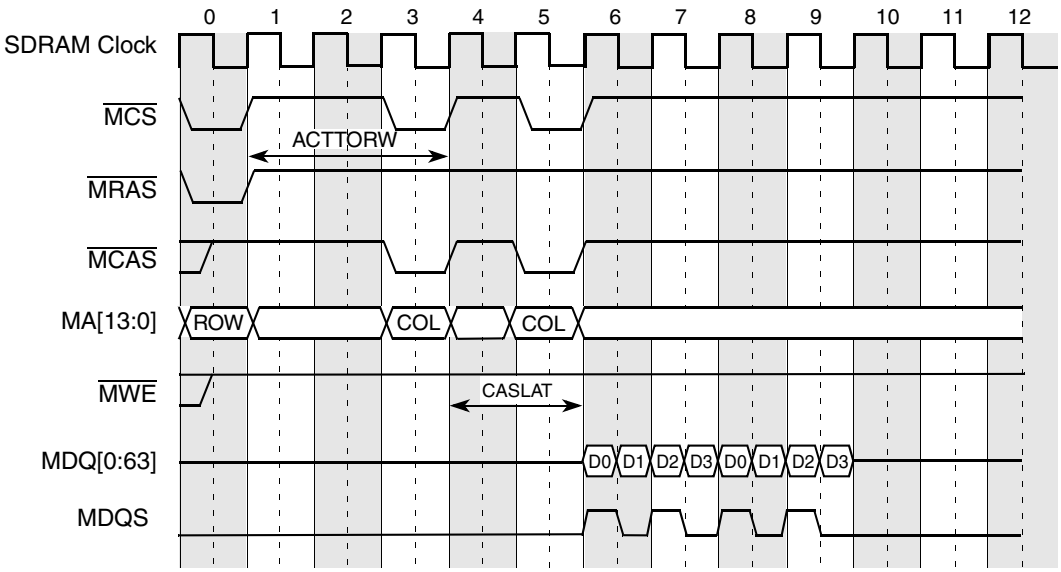


Figure 9-26. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

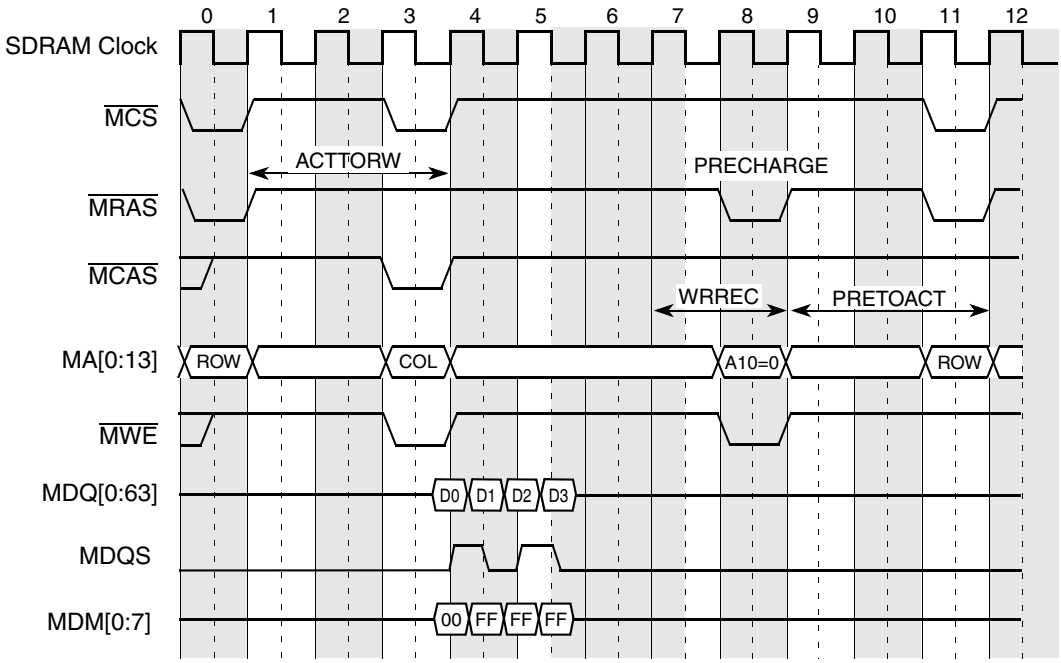


Figure 9-27. DDR SDRAM Single-Beat (Double-Word) Write Timing—ACTTORW = 3

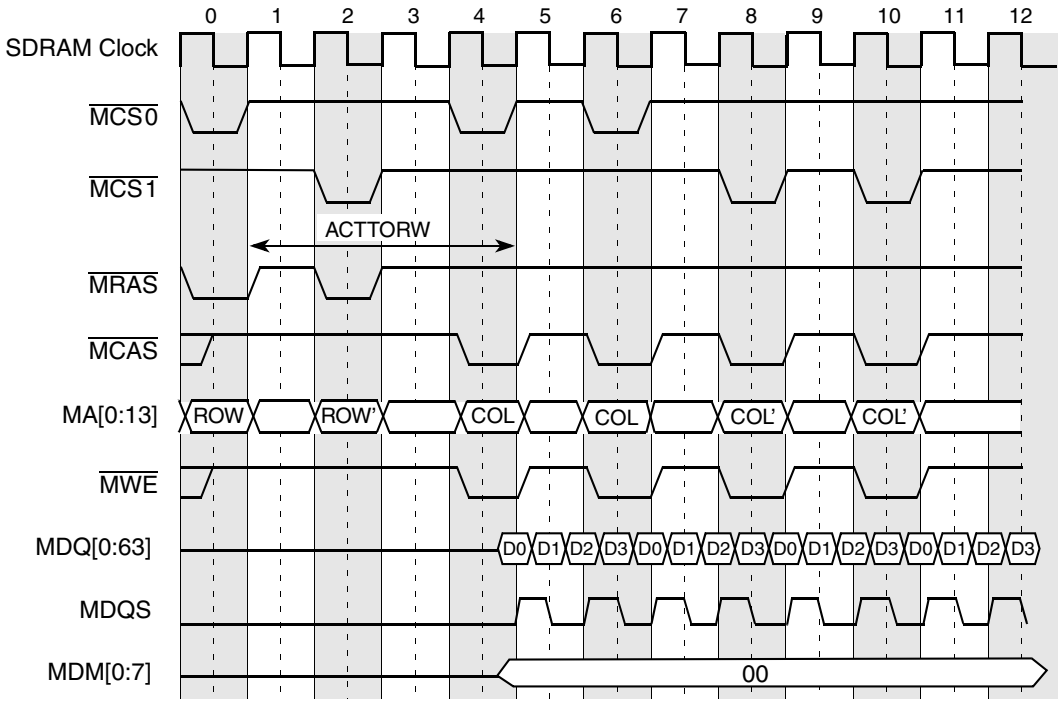


Figure 9-28. DDR SDRAM Burst Write Timing—ACTTORW = 4

9.5.4.1 Clock Distribution

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72 bit × 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.

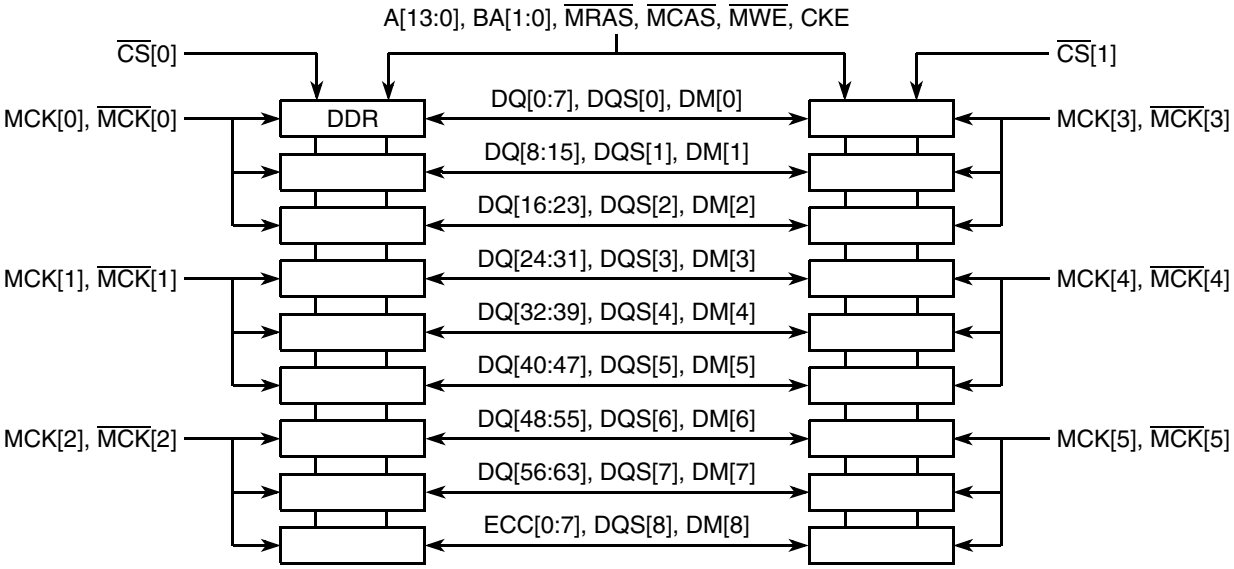


Figure 9-29. DDR SDRAM Clock Distribution Example

9.5.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the extended mode and base mode register data DDR_SDRAM_MODE[ESDMODE,SDMODE] to the SDRAM array by issuing two mode-set commands separated by two SDRAM clock periods. Figure 9-30 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. Following commands must wait two SDRAM cycles.

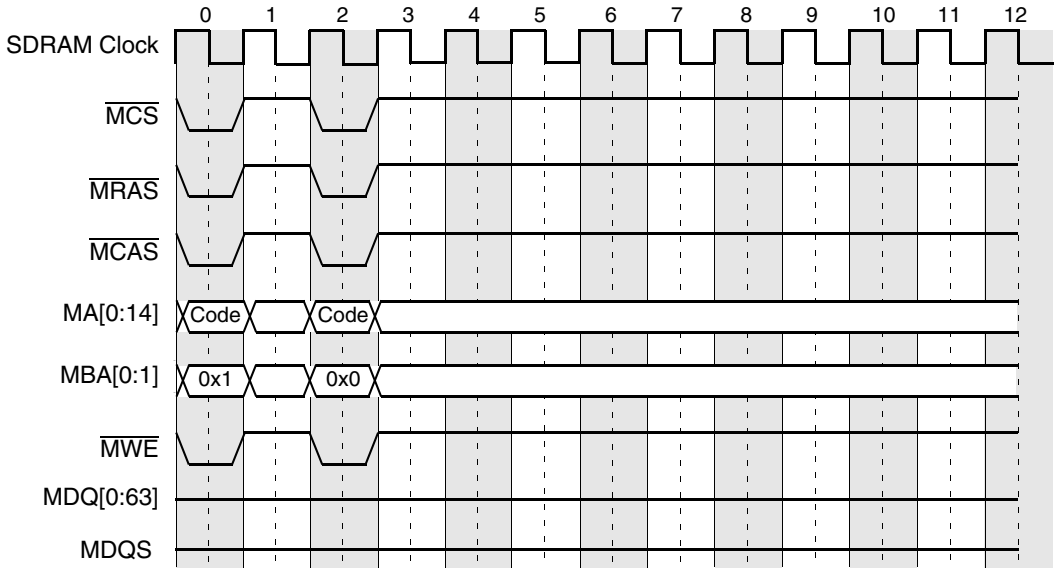


Figure 9-30. DDR SDRAM Mode-Set Command Timing

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.

Enabling registered DIMM mode does not affect bus timing for DDR reads. However to compensate for latch delay on the registered DIMMs' control signals, the programmed SDRAM read latency value (TIMING_CFG_1[CASLAT]) must be one greater than the value needed for non-registered DIMMs. Figure 9-31 shows the registered DDR SDRAM DIMM back-to-back burst write timing.

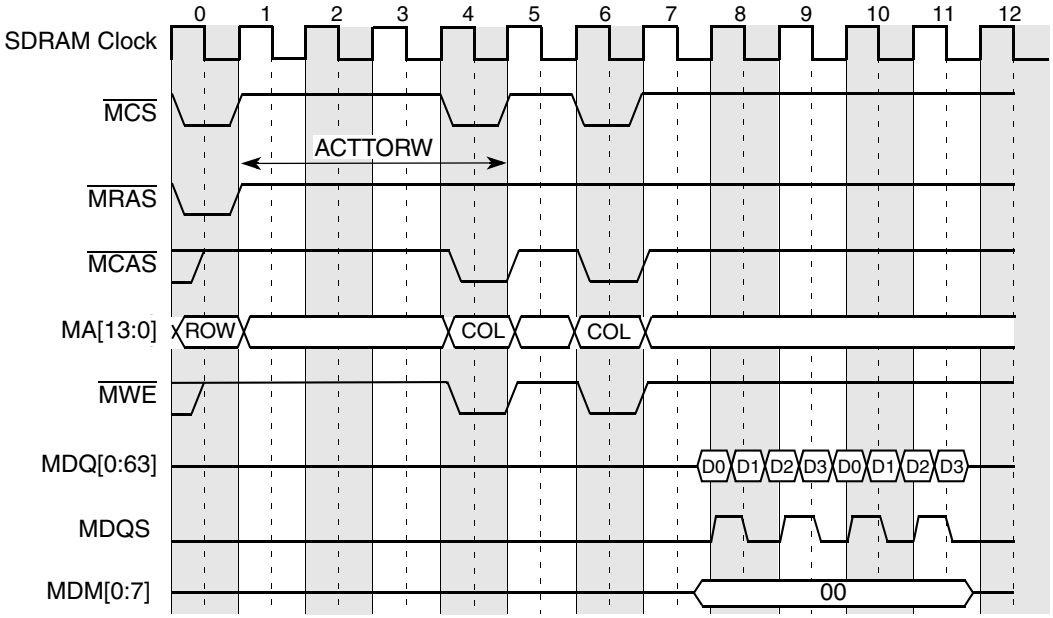


Figure 9-31. Registered DDR SDRAM DIMM Burst Write Timing

9.5.7 DDR SDRAM Source Synchronous Clock Control

The DDR memory controller provides the ability to compensate for system-specific timing conditions resulting from, for instance, trace length variations, or significant memory interface signal loading. The DDR_SDRAM_CLK CNTL register may be used to source synchronously delay launching of the MCK/MCK_B clock signals relative to address/command information. When enabled, the SDRAM clock may be shifted up to one full cycle within the address/command data valid eye with 1/4 cycle granularity. Refer to Section 9.4.1.8, "DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)," for register and configuration details.

9.5.8 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 1/4th SDRAM clock periods starting with the default value of 1/4th period delay. Figure 9-32 shows the use of the WR_DATA_DELAY parameter.

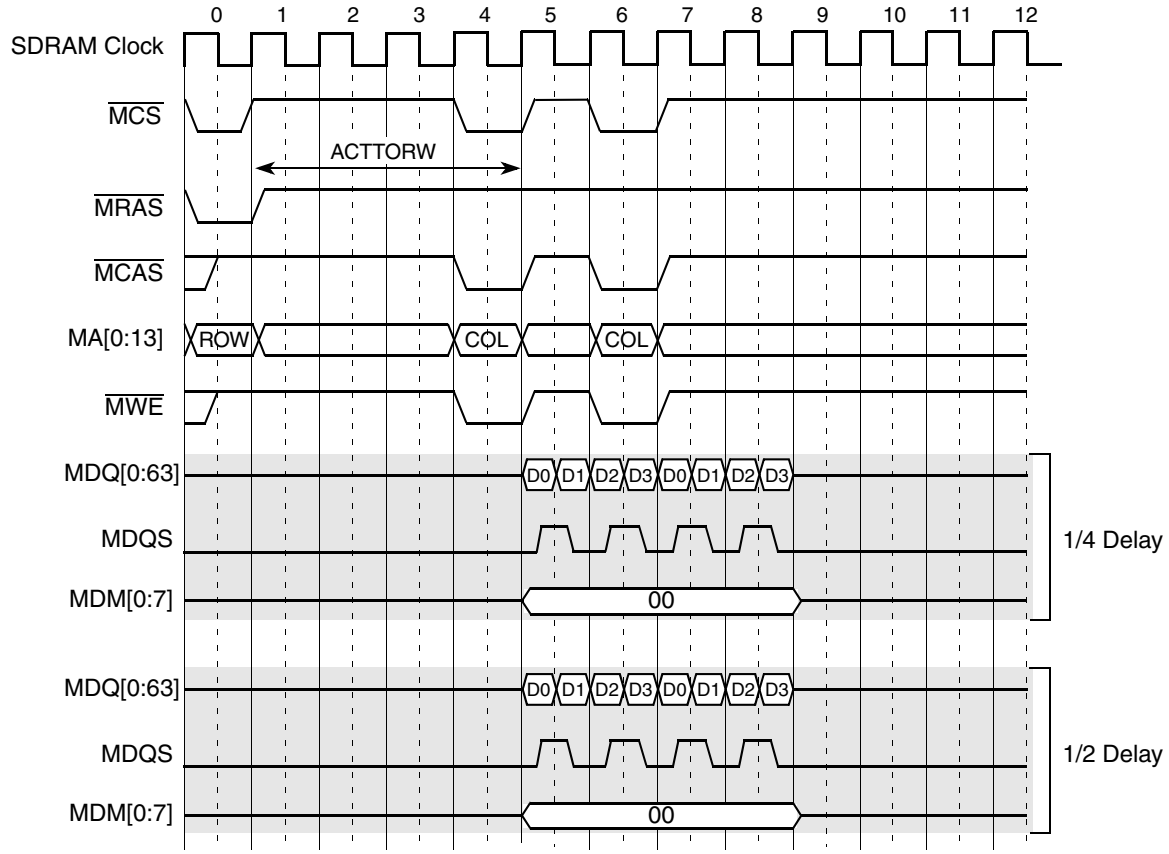


Figure 9-32. Write Timing Adjustments Example

9.5.9 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state or a soft-stop state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow 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 an auto-refresh command 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 must be 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. CKE is negated at this time, so it would not be possible to stagger two more refresh commands. 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].

9.5.9.1 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 9-33 (TIMING_CFG_1 [REFREC] = 10 in this example).

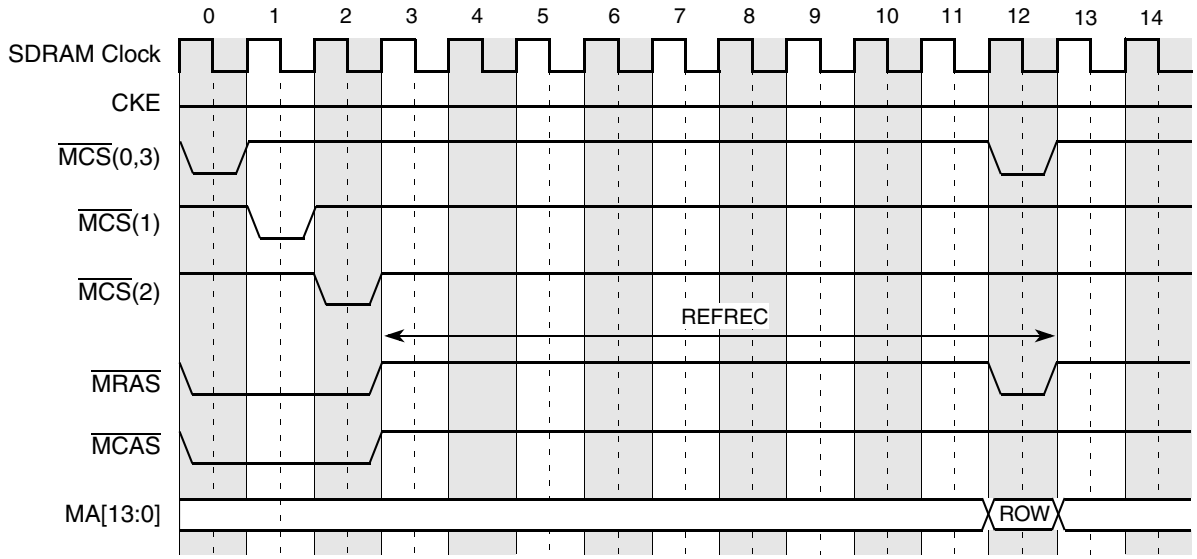


Figure 9-33. DDR SDRAM Bank-Staggered Auto-Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

9.5.9.2 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 of the DDR_SDRAM_CFG register. Table 9-31 summarizes the refresh types available in each power-saving mode.

Table 9-31. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	0

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.

All open pages are precharged before self refresh mode is entered.

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].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of DDR_SDRAM_INTERVAL[BSTOPRE]. Powering up the DDR SDRAM when a new memory reference is scheduled causes a one-clock access latency penalty, as shown in Figure 9-34.

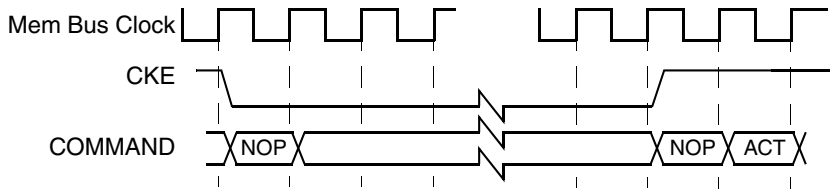


Figure 9-34. DDR SDRAM Power-Down Mode

9.5.9.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-35 and Figure 9-36.

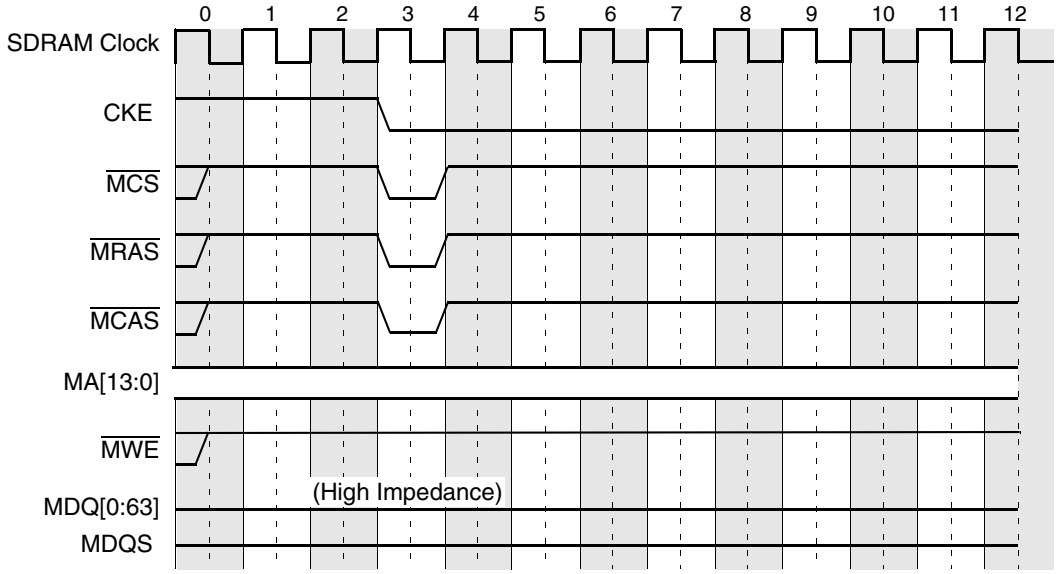


Figure 9-35. DDR SDRAM Self-Refresh Entry Timing

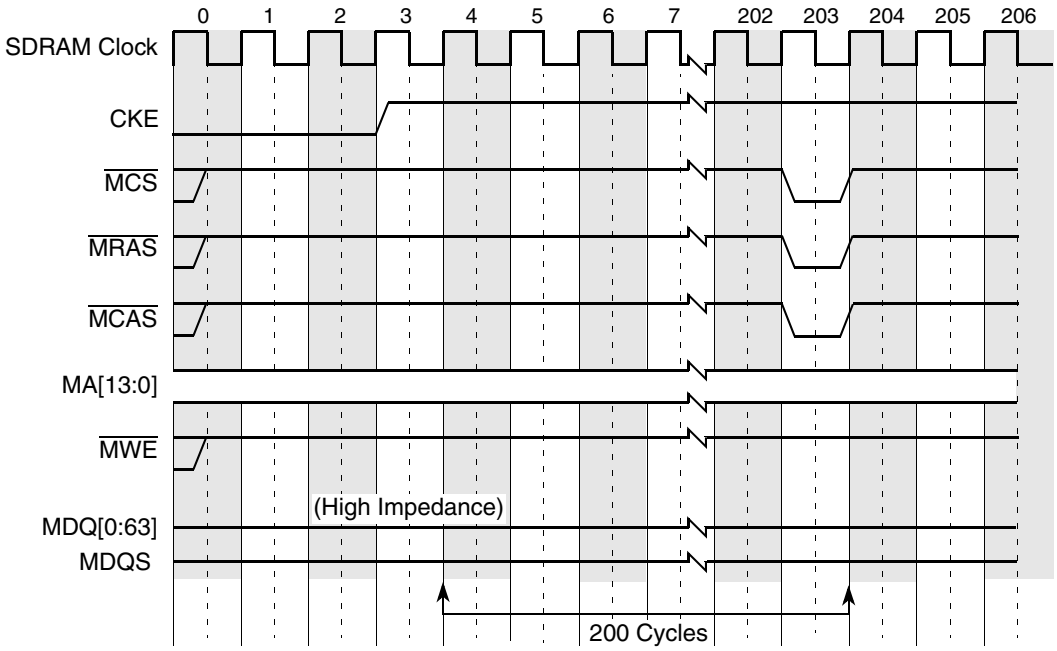


Figure 9-36. DDR SDRAM Self-Refresh Exit Timing

9.5.10 DDR Data Beat Ordering

Transfers to and from memory are always performed in four-beat bursts (four beats = 32 bytes). For transfer sizes other than four beats, the data transfers are still operated as four-beat bursts. If ECC is enabled for a sub-doubleword write transaction, a full read-modify-write is performed to properly update

ECC bits. If ECC is disabled then no read-modify-write is required for sub-doubleword writes, and the data masks (MDM[0:8]) are used to prevent writing 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.

Table 9-32 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 9-32. Memory Controller—Data Beat Ordering

Transfer Size	Starting Double-Word Offset	Double-Word Sequence[1] to/from DRAM and Queues
1 double word	0 1 2 3	**0** - 1 - 2 - 3 **1** - 2 - 3 - 0 **2** - 3 - 0 - 1 **3** - 0 - 1 - 2
2 double words	0 1 2	**0 - 1** - 2 - 3 **1 - 2** - 3 - 0 **2 - 3** - 0 - 1
3 double words	0 1	**0 - 1 - 2** - 3 **1 - 2 - 3** - 0
4 double words	0 1 2 3	**0 - 1 - 2 - 3** **1 - 2 - 3 - 0** **2 - 3 - 0 - 1** **3 - 0 - 1 - 2**

[1] All underlined double-word offsets are valid for the transaction.

9.5.11 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, the DDR memory controller uses the SDRAM AUTO PRECHARGE feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable AUTO PRECHARGE. AUTO PRECHARGE is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains opens until one of the following conditions occurs:

- Refresh interval is met
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks,

especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] and CS_n_CONFIG[AP_n_EN]

9.5.12 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. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, 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. This read-modify-write operation is performed as an atomic transaction in the DDR controller. The write command is then issued 3–5 memory clocks after the completion of the read, depending on various system parameters. 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 9-33 and Table 9-34.

Table 9-33. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		
8	•	•					•	
9	•		•				•	
10	•			•			•	
11	•				•		•	
12	•	•				•	•	•
13	•		•			•	•	•
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•
40			•	•			•	
41			•		•		•	
42	•		•		•		•	•
43		•	•		•		•	•
44			•	•		•	•	•
45			•		•	•	•	•

Table 9-33. 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
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 9-34. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•

9.5.13 Error Management

The DDR memory controller detects three different kinds of errors: 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 9.4.1.17, "Memory Error Interrupt Enable (ERR_INT_EN)," Section 9.4.1.16, "Memory Error Disable (ERR_DISABLE)," and Section 9.4.1.15, "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 an interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates an interrupt (if enabled, as described in Section 9.4.1.16, "Memory Error Disable (ERR_DISABLE)"). The final error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate an interrupt (if enabled, as described in Section 9.4.1.15, "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. Table 9-35 shows the errors with their descriptions.

Table 9-35. 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 an 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 $\overline{\text{MCS}}$[0:3] signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to multiplex appropriately the row and column address bits for each bank. Refer to row-address configuration in Section 9.4.1.2, "Chip Select Configuration (CSn_CONFIG)." Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (bootcode) must set up the programmable parameters in the memory interface configuration registers (MICRs). See Section 9.4.1, "Register Descriptions," for more detailed descriptions of the configuration registers. These parameters are shown in Table 9-36.

Table 9-36. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/Page
CS*n*_BNDS	Chip select memory bounds	SA*n* EA*n*	9.4.1.1/9-9
CS*n*_CONFIG	Chip select configuration	CS_*n*_EN ROW_BITS_CS_*n* COL_BITS_CS_*n*	9.4.1.2/9-10
TIMING_CFG_1	DDR SDRAM timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT REFREC WRREC ACTTOACT WRTORD WR_DATA_DELAY	9.4.1.3/9-11
DDR_SDRAM_CFG	DDR SDRAM control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR	9.4.1.5/9-13
DDR_SDRAM_MODE	DDR SDRAM mode configuration	ESDMODE SDMODE	9.4.1.6/9-14
DDR_SDRAM_INTERVAL	DDR SDRAM interval configuration	REFINT BSTOPRE	9.4.1.7/9-15

9.6.1 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CLK_CNTL[SS_EN] and DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs must elapse after the memory clocks are stable (that is, initialization is complete of all clock related configuration registers) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. After MEM_EN has been set, the DDR memory controller automatically performs the JEDEC-compliant initialization sequence to initialize memories according to the information in the SDMODE and ESDMODE fields of the DDR_SDRAM_MODE register. The initialization sequence is as follows:

1. PRECHARGE ALL
2. MODE REGISTER SET for extended mode register
3. MODE REGISTER SET for mode register
4. PRECHARGE ALL
5. Two AUTO REFRESH commands
6. MODE REGISTER SET for mode register with reset DLL bit deactivated

Note that the BA0 and BA1 bits are automatically driven appropriately during the MODE REGISTER SET commands. After this automatic initialization is complete the memory array is ready for access and the memory controller begins processing memory transactions as they arrive.

Chapter 10
Programmable Interrupt Controller

This chapter describes the programmable interrupt controller (PIC) interrupt protocol, various types of interrupt sources controlled by the PIC unit, and the PIC registers with some programming guidelines.

10.1 Introduction

A block diagram of portions of the MPC8555E showing the relationship of the various functional blocks and external signals to the PIC unit is shown in Figure 10-1.

The MPC8555E PIC unit prioritizes and manages interrupts from the L2 cache, the ECM, the DDR controller, the local bus controller (LBC), the 4-channel DMA controller, the PCI block, the dual three-speed Ethernet controllers (TSECs), the DUART, the CPM, the performance monitor, and the I^2C controller. It also manages the interrupts generated by the PIC itself and by off-chip interrupt sources.

The PIC unit receives interrupt signals from three sources: external to the integrated device, internal to the integrated device, and intrinsic to the PIC itself. The PIC selects among all current interrupts the one with the highest priority and forwards it to the internal processor core, or, in pass-through mode, off-chip for external servicing.

10.1.1 Overview

The PIC is compliant with the OpenPIC architecture. The interrupt controller provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to the CPU for servicing.

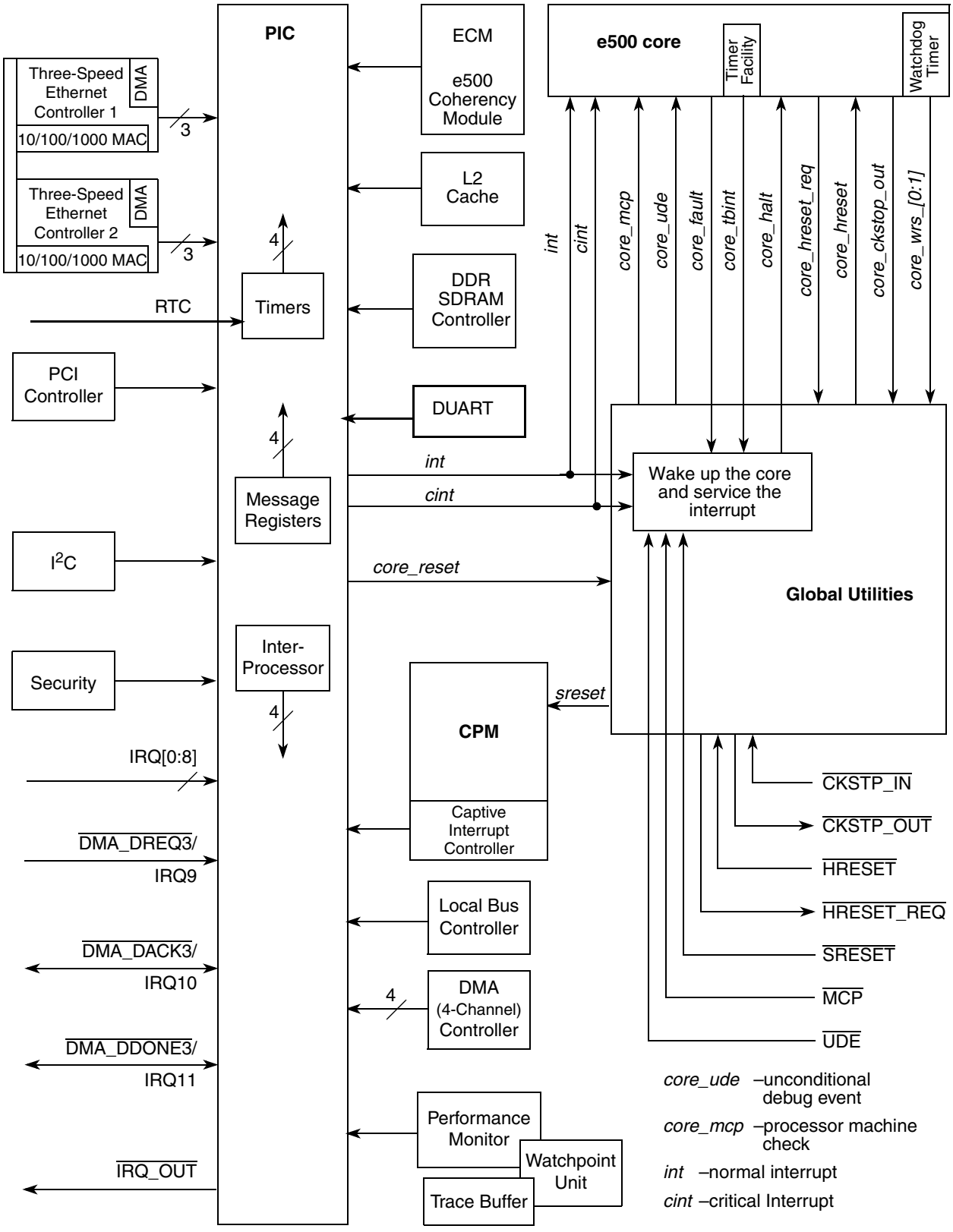


Figure 10-1. MPC8555E Interrupt Sources Block Diagram

10.1.2 Features

- Programming model compliant with the OpenPIC architecture
- Support for 12 external and 21 internal interrupt sources. Serial interrupts are not supported.
- Four interprocessor interrupt channels
- Four 32-bit messaging interrupt channels
- Four global high resolution timers that can be clocked with the CCB (platform) clock or the RTC input.
- Fully-nested interrupt delivery
- Processor initialization control
- Programmable resetting of the PIC unit through the global configuration register
- 16 programmable interrupt priority levels
- Support for connection of external interrupt controller device such as an 8259 programmable interrupt controller
- In 8259 mode, it generates a local (internal) interrupt output signal, $\overline{\text{IRQ\_OUT}}$.
- Recovery from spurious interrupts

10.1.3 Interrupts to the Processor Core

The *int* signal, which causes the external interrupt exception, is the main interrupt output from the PIC unit to the processor core. Interrupts can alternately be configured as critical interrupts (in the destination registers); these are reported to the core through the *cint* signal. The Book E architecture implemented by the e500 core defines a separate critical interrupt type with its own save and restore registers (CSRR0 and CSRR1) and return instruction (Return from Critical Interrupt, **rfci**). In addition to the external and critical interrupts that are generated by the PIC, other MPC8555E conditions (shown in Table 10-1) cause interrupts to the core (and wake up the core when it is in a low-power state). Note that interrupts from port C of the CPM are a special case and do not reach the PIC when the device is asleep. Therefore, they do not cause the device to wake up.

Table 10-1. Processor Interrupts Generated Outside the Core—Types and Sources

Core Interrupt Type	Signaled by (Input to Core)	Sources
PIC-Programmable Interrupts		
External interrupt	*int*	Generated by the PIC, as described in Section 10.1.5, “Interrupt Sources.”
Critical interrupt	*cint*	Generated by the PIC, as described in Section 10.1.5, “Interrupt Sources.”
Other Interrupts Generated Outside the Core		
Machine check	*core_mcp* (MPC8555E causes)	• $\overline{\text{MCP}}$ • $\overline{\text{SRESET}}$ • Assertion of *core_mcp* by global utilities block

Table 10-1. Processor Interrupts Generated Outside the Core—Types and Sources (continued)

Core Interrupt Type	Signaled by (Input to Core)	Sources
Unconditional debug event	*core_ude*	$\overline{\text{UDE}}$. Asserting $\overline{\text{UDE}}$ generates an unconditional debug exception type debug interrupt and sets a bit in the debug status register, DBSR[UDE], as described in Section 6.13.2, "Debug Status Register (DBSR)."
Reset	*core_hreset*	• $\overline{\text{HRESET}}$ assertion (and negation) • *core_hreset_req*. Internal signal to the device, output from core, input to the platform—caused by writing to the core DBCR0[RST]. This condition is additionally qualified with MSR[DE] and DBCR0[IDM] bits. Note that assertion of this signal causes a hard reset of the core only. • *core_hreset_req* can also be caused by a second timer timeout condition as described in Section 6.6.1, "Timer Control Register (TCR)." • *core_reset*. Output from PIC. See Section 10.3.1.4, "Processor Initialization Register (PIR)."

The global utilities block monitors two additional interrupt conditions generated by the e500 core (*core_tbint* and *core_fault_out* signals), as shown in Table 10-2. Assertion of either of these signals causes the processor to exit a low-power state. These cases are caused by core conditions, and after the global utilities logic wakes up the core, they are handled by the core as shown in Table 10-2.

Table 10-2. e500 Core-Generated Interrupts That Cause a Wake-Up

Core Interrupt Type	Signaled by (Output from Core)	Sources
Fixed interval timer	*core_tbint*	The source of both of these interrupts is the time base facility within the e500 core. The MPC8555E monitors this core output signal and considers it a processor interrupt for the purposes of power management (causes the core to exit a low-power state). For more information about the interaction between core-generated signals and power management, see Chapter 18, "Global Utilities."
Decrementer		
Machine check	*core_fault_out*	Occurs when the L1 cache has a parity error on a snoop push operation, which can occur while the core is halted. The MPC8555E monitors this signal and considers it a processor interrupt for the purposes of power management (causes the core to exit a low-power state).

10.1.4 Modes of Operation

Mixed or pass-through mode can be chosen by setting or clearing GCR[M] as described in Section 10.3.1.2, "Global Configuration Register (GCR)."

10.1.4.1 Mixed Mode (GCR[M] = 1)

In mixed mode, the external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in Section 10.3.6.1, "External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)," through Section 10.3.6.4, "Internal Interrupt Destination Registers (IIDR0–IIDR31)."

10.1.4.2 Pass-Through Mode (GCR[M] = 0)

The PIC unit provides a mechanism to support alternate external interrupt controllers such as the PC/AT compatible 8259 interrupt controller architecture. After a hard reset, the PIC unit defaults to pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to the e500 core, as shown in Table 10-2, all other external interrupt signals are ignored. Thus, the interrupt signal from an external interrupt controller can be connected to IRQ0 and cause direct interrupts to the processor. The PIC does not perform a vector fetch from an 8259 interrupt controller.

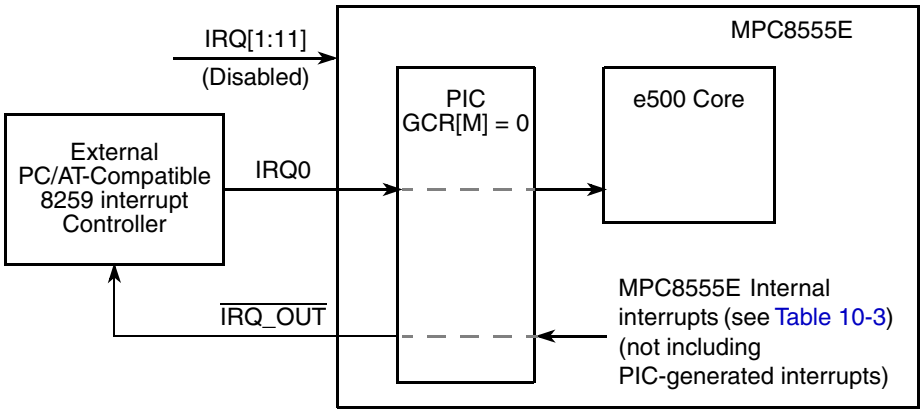


Figure 10-2. Pass-Through Mode Example

When pass-through mode is enabled, the internally-generated interrupts shown in Table 10-3, are not forwarded to the e500 core. Instead, the PIC passes the raw interrupts from the internal sources to $\overline{\text{IRQ\_OUT}}$.

Note that in pass-through mode, interrupts generated by the PIC itself (global timers, interprocessor, and message register interrupts) cannot be used. If internal or PIC-generated interrupts must be reported internally to the processor, pass-through mode must be disabled.

10.1.5 Interrupt Sources

Aside from the sources of machine check, unconditional debug event, and reset interrupts to the core described in Table 10-1, the PIC unit can receive 45 separate interrupts from five different sources as follows:

- 12 external—Off-chip signals, IRQ[0:11]
- 21 internal—On-chip. Sources are L2, ECM, DDR, LBC, DMA, PCI, TSEC, DUART, CPM, performance monitor, and I^2C.
- Four global timers—From inside the PIC
- 4 inter-processor (IPI)—Intended for communication between different processor cores on the same device. Only used for self-interrupt in a single-core device such as the MPC8555E.
- 4 message registers—From inside the PIC. Triggered on register write, cleared on read. Used for inter-process communication.

10.1.5.1 Interrupt Routing—Mixed Mode

When the PIC receives an internal or external interrupt, its destination register is checked to determine if it should be routed off-chip to the external $\overline{\text{IRQ\_OUT}}$ signal (if the incoming interrupt has its xIDR[EP] bit set). Alternatively, if the incoming interrupt has been configured in xIDR[CI] as a critical interrupt, the PIC completes its processing of the interrupt by asserting the *cint* input to the core, causing it to be serviced as a critical interrupt. As a third alternative (if neither xIDR[CI] or xIDR[EP] are set), the interrupt can be serviced as a normal external interrupt by the processor core (through the *int* signal). In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow described in Section 10.4.1, "Flow of Interrupt Control," is followed.

10.1.5.2 Internal Interrupt Sources

Table 10-3 shows the assignments of the 21 internal interrupt sources for the MPC8555E. Note that this list does not include the interrupts generated by the PIC unit.

Table 10-3. Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source	Internal Interrupt Number	Interrupt Source
0	L2 cache	15–17	Reserved
1	ECM	18	TSEC 1 receive/transmit error interrupt
2	DDR DRAM	19	TSEC 2 transmit interrupt
3	LBC	20	TSEC 2 receive interrupt
4	DMA channel 0	21–23	Reserved
5	DMA channel 1	24	TSEC 2 receive/transmit error interrupt
6	DMA channel 2	25	Reserved
7	DMA channel 3	26	DUART
8	PCI1	27	I^2C controller
9	PCI2	28	Performance monitor interrupt
10–12	Reserved	29	Security
13	TSEC 1 transmit interrupt	30	CPM (note that interrupts from port C are not signaled to the PIC when the device is asleep)
14	TSEC 1 receive interrupt	31	Reserved

10.2 External Signal Description

The following sections provide an overview and detailed descriptions of the PIC signals.

10.2.1 Signal Overview

PIC interface signals are described in Table 10-4. There are 12 distinct external interrupt request input signals (IRQ[0:11]) and 1 interrupt request output signal (IRQ_OUT). As Table 10-4 shows, three IRQ inputs are multiplexed with DMA signals for DMA channel 3.

Table 10-4. PIC Interface Signals

Signal Name	I/O	Description
IRQ[0:8]	I	External interrupts
IRQ9/DMA_DREQ3[1]	I	External interrupt/DMA channel 3 request
IRQ10/DMA_DACK3[1]	I or O	External interrupt (input)/DMA channel 3 acknowledge (output)
IRQ11/DMA_DDONE3[1]	I or O	External interrupt (input)/DMA channel 3 done (output)
IRQ_OUT	O	Interrupt request out
MCP	I	Processor machine check
UDE	I	Unconditional debug event

1 IRQ9–IRQ11 are multiplexed with DMA3 signals. These functions are mutually exclusive; the active function is specified in PMUXCR of the global utilities block as described in Section 18.4.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

10.2.2 Detailed Signal Descriptions

Table 10-5 provides detailed descriptions of the external PIC signals.

Table 10-5. Interrupt Signals—Detailed Signal Descriptions

Signal	I/O	Description	
IRQ[0:11]	I	Interrupt request 0–11. The polarity and sense of each of these signals is programmable. All of these inputs can be driven completely asynchronously.	
		State Meaning	Asserted—When an external interrupt signal is asserted (according to the programmed polarity), the priority is checked by the PIC unit, and the interrupt is conditionally passed to the processor. In pass-through mode, only interrupts detected on IRQ0 are passed directly to the processor core. 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 10-5. Interrupt Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{IRQ\_OUT}}$	O	Interrupt request out. Active-low, open drain. When the PIC is programmed in pass-through mode, this output reflects the raw interrupts generated by on-chip sources. See Section 10.1.4, "Modes of Operation," for more details.	
		State Meaning	Asserted—At least one interrupt is currently being signaled to the external system. Negated—Indicates no interrupt source currently routed to $\overline{\text{IRQ\_OUT}}$.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of $\overline{\text{IRQ\_OUT}}$ occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 CCB clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Message interrupts: 2 cycles after write to message register. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 2 CCB clock cycles External interrupt: 4 cycles. Message interrupts: 2 cycles after message register cleared.
$\overline{\text{MCP}}$	I	Machine check processor. Assertion causes a machine check interrupt to the e500 core. Note that if the e500 core is not configured to process machine check interrupts (MSR[ME] = 0), assertion of $\overline{\text{MCP}}$ causes a checkstop condition. Note that internal sources for the internal *core_mcp* signal can also cause a machine check interrupt to the processor core, as described in Section 18.4.1.13, "Machine Check Summary Register (MCPSUMR)," Table 10-1 and Table 10-2.	
		State Meaning	Asserted—The MPC8555E should initiate a machine check interrupt or enter the checkstop state as directed by the MSR. Negated—Machine check handling is not being requested by the external system.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Because $\overline{\text{MCP}}$ is edge-triggered, it can be negated one clock after its assertion.
$\overline{\text{UDE}}$	I	Unconditional debug event. Assertion signal causes an unconditional debug exception to the e500 core.	
		State Meaning	Asserted—Indicates that the MPC8555E should initiate an unconditional debug event interrupt to the processor core. Negated—Indicates that unconditional debug event handling is not being requested by $\overline{\text{UDE}}$.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Should remain asserted until software in the unconditional debug event interrupt handler causes the external device asserting the $\overline{\text{UDE}}$ signal to negate it.

10.3 Memory Map/Register Definition

The PIC programmable register map occupies 256 Kbytes of memory-mapped space. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All PIC registers are 32 bits wide and, although located on 128-bit address boundaries, should only be accessed as 32-bit quantities.

The PIC address offset map, shown in Table 10-6, is divided into the following three areas:

- 0x*nn*4_0000–0x*nn*4_FFF0—Global registers
- 0x*nn*5_0000–0x*nn*5_FFF0—Interrupt source configuration registers
- 0x*nn*6_0000–0x*nn*7_FFF0—Per-CPU registers

Table 10-6. PIC Register Address Map

Offset	Register	Access	Reset	Section/Page
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.7.1/10-37
0x4_0050	IPIDR1—IPI 1 dispatch register			
0x4_0060	IPIDR2—IPI 2 dispatch register			
0x4_0070	IPIDR3—IPI 3 dispatch register			
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.7.2/10-38
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.7.3/10-39
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.7.4/10-39
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.7.5/10-40
0x4_00C–0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x0037_0002	10.3.1.1/10-15
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.2/10-15
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.3/10-16
0x4_1090	PIR—Processor initialization register	R/W	0x0000_0000	10.3.1.4/10-16
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.5/10-17
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.6/10-18
0x4_10F0	TFRR—Timer frequency reporting register	R/W	0x0000_0000	10.3.2.1/10-19
0x4_1100	GTCCR0—Global timer 0 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_1110	GTBCR0—Global timer 0 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_1120	GTVPR0—Global timer 0 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_1130	GTDR0—Global timer 0 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_1140	GTCCR1—Global timer 1 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_1150	GTBCR1—Global timer 1 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_1160	GTVPR1—Global timer 1 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1170	GTDR1—Global timer 1 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_1180	GTCCR2—Global timer 2 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_1190	GTBCR2—Global timer 2 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_11A0	GTVPR2—Global timer 2 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_11B0	GTDR2—Global timer 2 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_11C0	GTCCR3—Global timer 3 current count register	R	0x0000_0000	10.3.2.2/10-19
0x4_11D0	GTBCR3—Global timer 3 base count register	R/W	0x8000_0000	10.3.2.3/10-20
0x4_11E0	GTVPR3—Global timer 3 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-20
0x4_11F0	GTDR3—Global timer 3 destination register	R/W	0x0000_0001	10.3.2.5/10-21
0x4_1200– 0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-22
0x4_1310	IRQSR0—$\overline{\text{IRQ\_OUT}}$ summary register 0	R	0x0000_0000	10.3.3.1/10-24
0x4_1320	IRQSR1—$\overline{\text{IRQ\_OUT}}$ summary register 1	R	0x0000_0000	10.3.3.2/10-25
0x4_1330	CISR0—Critical Interrupt summary register 0	R	0x0000_0000	10.3.3.3/10-26
0x4_1340	CISR1—Critical Interrupt summary register 1	R	0x0000_0000	10.3.3.4/10-26
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-27
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-28
0x4_13D0– 0x4_13F0	Reserved	—	—	—
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-28
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1440– 0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-29
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-29

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1520–0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-30
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	10.3.6.2/10-31
0x5_0180–0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	10.3.6.4/10-33

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0550	IIDR126—Internal interrupt 26 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-32
0x5_05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	10.3.6.4/10-33
0x5_0600– 0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	10.3.6.6/10-35

Table 10-6. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-34
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	10.3.6.6/10-35
0x5_1680–0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per-CPU Registers				
0x6_0000–0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—P0 IPI 0 dispatch register	W	All zeros	10.3.7.1/10-37
0x6_0050	IPIDR1—P0 IPI 1 dispatch register			
0x6_0060	IPIDR2—P0 IPI 2 dispatch register			
0x6_0070	IPIDR3—P0 IPI 3 dispatch register			
0x6_0080	CTPR0—P0 current task priority register	R/W	0x0000_000F	10.3.7.2/10-38
0x6_0090	WHOAMI0—P0 who am I register	R	All zeros	10.3.7.3/10-39
0x6_00A0	IACK0—P0 interrupt acknowledge register	R	All zeros	10.3.7.4/10-39
0x6_00B0	EOI0—P0 end of interrupt register	W	All zeros	10.3.7.5/10-40

10.3.1 Global Registers

Most registers have one address. Some registers are replicated for each processor in a multiprocessor device. In this case, each processor accesses its separate registers using the same address, the address decoding being sensitive to the processor ID. A copy of the per-CPU registers is available to each processor core at the same physical address, that is, the private access address space. The private access address space acts like an alias to a processor's own copy of the per-CPU registers. As shown in Figure 10-31, the ID of the processor initiating the read/write transaction is used to determine which processor's per-CPU registers to access. For more information on per-CPU registers, see Section 10.3.7, "Per-CPU Registers."

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

10.3.1.1 Feature Reporting Register (FRR)

The feature reporting register (FRR) shown in Figure 10-3 provides information about interrupt and processor configurations. It also informs the programming environment of the controller version.

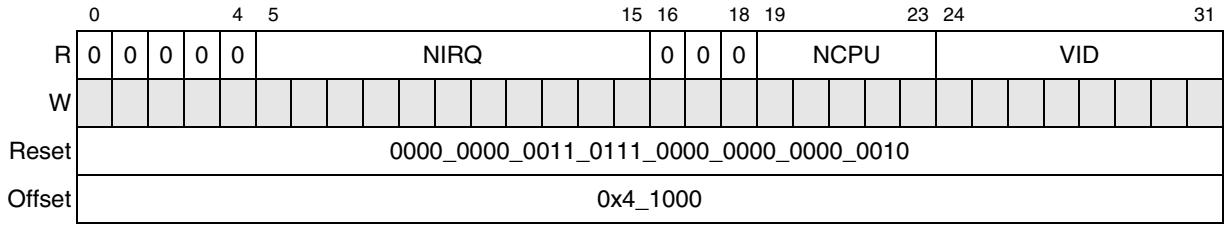


Figure 10-3. Feature Reporting Register (FRR)

Table 10-7 describes the FRR fields.

Table 10-7. FRR Field Descriptions

Bits	Name	Description
0–4	—	Reserved.
5–15	NIRQ	Number of interrupts. Contains the binary value of 1 minus the maximum number of interrupt sources supported. The value is 55 (0x37) because this device supports 56 interrupts: 12 external sources, 32 internal sources (only 21 used), four timer sources, four IPI sources and four messaging sources. A zero in this field corresponds to one source.
16–18	—	Reserved.
19–23	NCPU	Number of CPUs. The number of the highest physical CPU supported. The MPC8555E implements one CPU (the processor core), referenced as P0.
24–31	VID	Version ID. Version ID for this interrupt controller. Reports the OpenPIC specification revision level supported by this implementation. A value of two corresponds to revision 1.2, which is the revision level currently supported.

10.3.1.2 Global Configuration Register (GCR)

The global configuration register (GCR) shown in Figure 10-4 controls the PIC's operating mode, and allows software to reset the PIC.

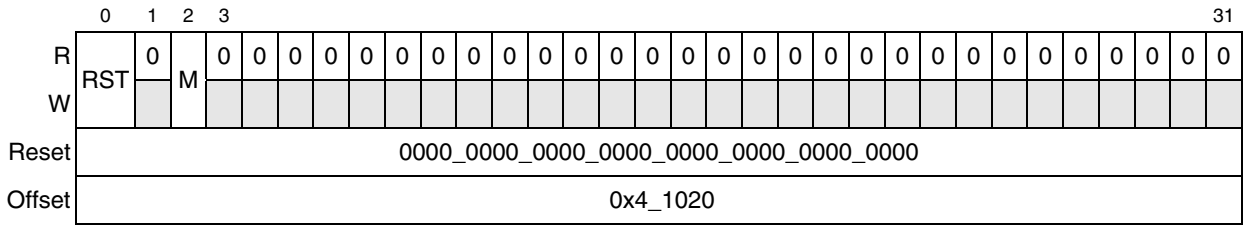


Figure 10-4. Global Configuration Register (GCR)

Table 10-8 describes the GCR fields.

Table 10-8. GCR Field Descriptions

Bits	Name	Description
0	RST	Reset. Setting this field forces the PIC to be reset. Cleared automatically when the reset sequence is complete. See Section 10.4.7, "Reset of the PIC," for more information on resetting the PIC.
1	—	Reserved
2	M	Mode. PIC operating mode. 0 Pass-through mode. On-chip PIC is disabled and interrupts detected on IRQ0 are passed directly to the processor core. See Section 10.1.4, "Modes of Operation," for more details. 1 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the PIC. See Section 10.1.4, "Modes of Operation," for more details.
3–31	—	Reserved

10.3.1.3 Vendor Identification Register (VIR)

The vendor identification register (VIR) shown in Figure 10-5 has specific read-only information about the vendor and the device revision.

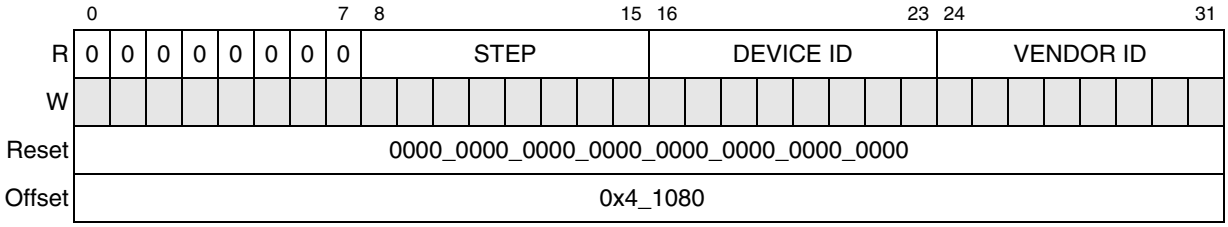


Figure 10-5. Vendor Identification Register (VIR)

Table 10-9 describes the VIR fields.

Table 10-9. VIR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STEP	Stepping. Indicates the silicon revision for this device. Has no meaning when the VENDOR ID value is zero.
16–23	DEVICE ID	Device identification. Vendor-specified identifier for this device. Has no meaning when the VENDOR ID value is zero.
24–31	VENDOR ID	Vendor identification. Specifies the manufacturer of this part. A value of zero implies a generic PIC-compliant device.

10.3.1.4 Processor Initialization Register (PIR)

The processor initialization register in the PIC provides a mechanism for software to reset the processor. The *core_reset* signal to the processor is held active until a zero is written to the processor initialization register field. Thus, PIR should only be written by an external host and the external host should wait at least 100 µsec to clear this field after writing it. The processor should not attempt to write to PIR because

writing to it resets the core, and the core will not be able to clear the reset field, and so it would remain in reset indefinitely.

Note that although the architecture was designed to support multiple processing cores, only fields corresponding to processors on the device are implemented, as shown in Figure 10-6. In a uniprocessor system, such as the MPC8555E, only PIR[P0] is implemented.

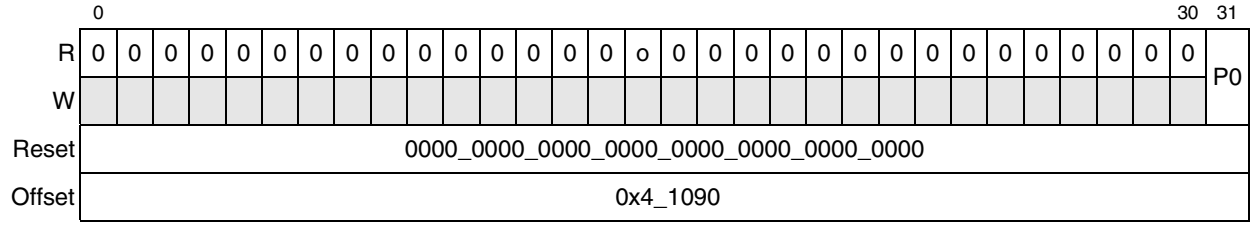


Figure 10-6. Processor Initialization Register (PIR)

Table 10-10 describes the PIR fields.

Table 10-10. PIR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0 core reset. Setting this bit causes the PIC unit to assert the *core_reset* signal to processor 0 (the e500 core).

10.3.1.5 IPI Vector/Priority Registers (IPIVPR*n*)

The interprocessor interrupt (IPI) vector/priority registers contain the interrupt vector and priority fields for the four interprocessor interrupt channels as shown in Figure 10-7. There is one IPI vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while IPIVPR*n*[A] is set.

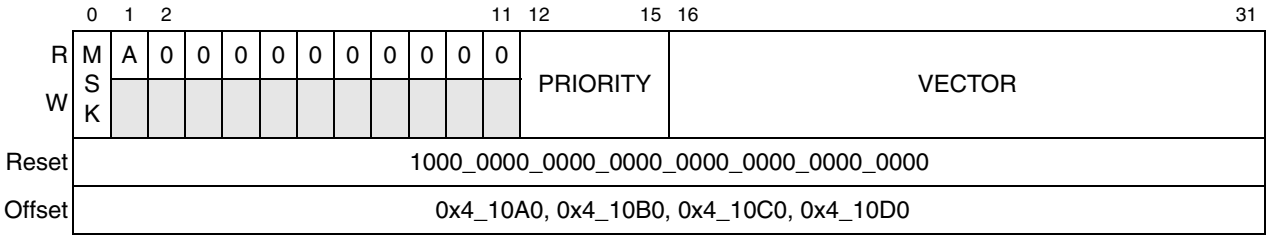


Figure 10-7. IPI Vector/Priority Register (IPIVPR*n*)

Table 10-11 describes the IPIVPR*n* fields.

Table 10-11. IPIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. Always set following reset. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been reported or is in-service. Note that this field is read only. The VECTOR and PRIORITY values should not be changed while IPIVPR*n*[A] is set. 0 No current interrupt activity associated with this source. 1 The interrupt bit for this source in the IPR or ISR is set.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupt reporting from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in Figure 10-37.

10.3.1.6 Spurious Vector Register (SVR)

The spurious vector register (SVR) shown in Figure 10-8 contains the 16-bit vector returned to the processor when the IACK register is read for a spurious interrupt.

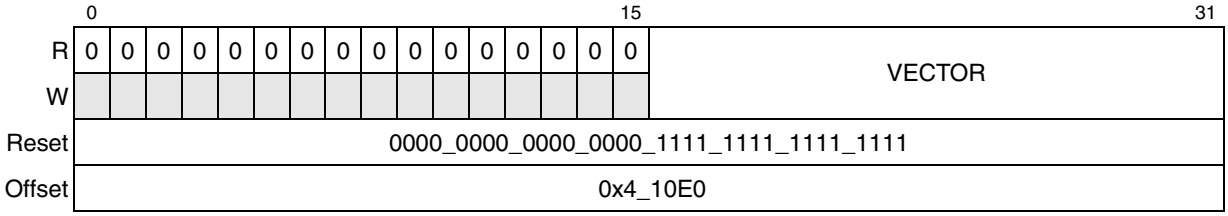


Figure 10-8. Spurious Vector Register (SVR)

Table 10-12 describes the SVR fields.

Table 10-12. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Spurious interrupt vector. Value returned when the interrupt acknowledge (IACK) register is read during a spurious vector fetch. See Section 10.4.4, “Spurious Vector Generation,” for more information about the events and conditions that may cause a spurious vector fetch.

10.3.2 Global Timer Registers

This section describes the global timer registers. Note that each of the four timers have four individual configuration registers (GTCCR*n*, GTBCR*n*, GTVPR*n*, GTDR*n*), but they are only shown once in this section.

10.3.2.1 Timer Frequency Reporting Register (TFRR)

The timer frequency reporting register (TFRR), shown in Figure 10-9, is written by software to report the clocking frequency of the PIC timers. Note that although TFRR is read/write, the value of this register is ignored by the PIC unit.

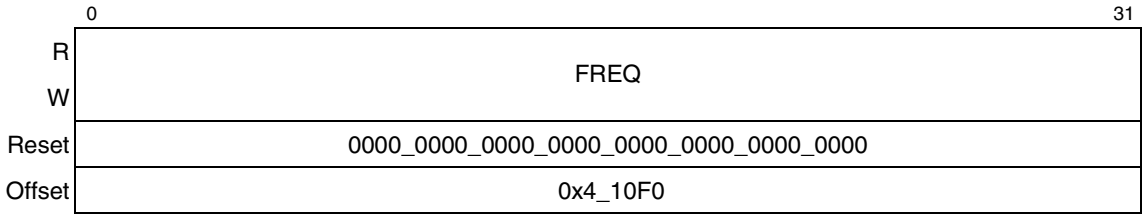


Figure 10-9. Timer Frequency Reporting Register (TFRR)

Table 10-13 describes the TFRR register.

Table 10-13. TFRR Field Descriptions

Bits	Name	Description
0–31	FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source, (identically the core complex bus (CCB) clock, to user software. See Section 4.4.4, "Clocking," for more details. TFRR is set only by software for later use by other applications and its value in no way affects the operating frequency of the global timers. The timers operate at a ratio of this clock frequency set by TCR[CLKR]. See Section 10.3.2.6, "Timer Control Register (TCR)."

10.3.2.2 Global Timer Current Count Registers (GTCCR*n*)

The global timer current count registers (GTCCRs), shown in Figure 10-10, contain the current count for each of the four PIC timers.

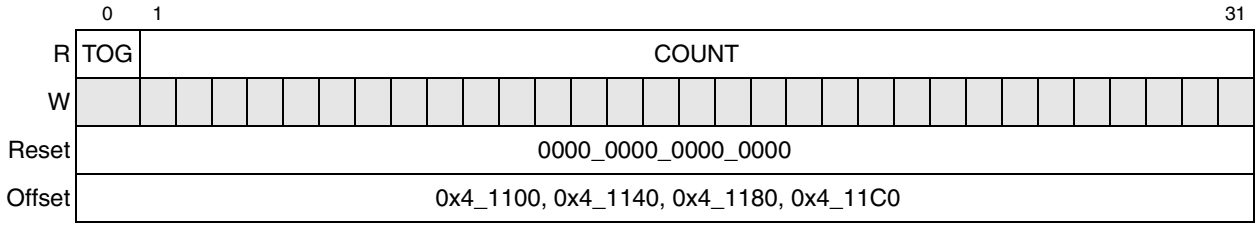


Figure 10-10. Global Timer Current Count Registers (GTCCR*n*)

Table 10-14 describes the GTCCR*n* fields.

Table 10-14. GTCCR*n* Field Descriptions

Bits	Name	Description
0	TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR*n*[CI] goes from 1 to 0.
1–31	COUNT	Current count. Decremented while GTBCR*n*[CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding base count register. Cascaded timers are reloaded with either all ones, or the contents of the base count register, depending on the value of TCR[ROVR]. See Section 10.3.2.6, “Timer Control Register (TCR),” for more details.

10.3.2.3 Global Timer Base Count Registers (GTBCR*n*)

The global timer base count registers (GTBCRs) contain the base counts for each of the four PIC timers as shown in Figure 10-11. This value is reloaded into the corresponding GTCCR*n* when the current count reaches zero. Note that when zero is written to the base count field, (and GTCCR*n*[CI] = 0), the timer generates an interrupt on every timer cycle.

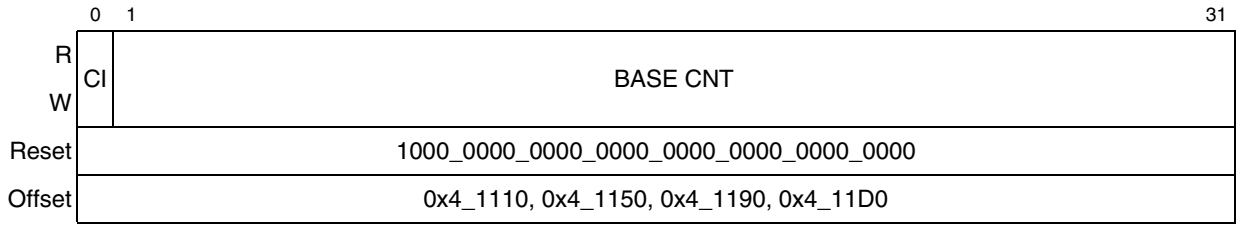


Figure 10-11. Global Timer Base Count Register (GTBCR*n*)

Table 10-15 describes the GTBCR*n* fields.

Table 10-15. GTBCR*n* Field Descriptions

Bits	Name	Description
0	CI	Count inhibit. Always set following reset 0 Counting enabled 1 Counting inhibited
1–31	BASE CNT	Base count. When CI transitions from 1 to 0, this value is copied into the corresponding current count register and the toggle bit is cleared. If CI is already cleared (counting is in progress), the base count is copied to the current count register at the next zero crossing of the current count.

10.3.2.4 Global Timer Vector/Priority Registers (GTVPR*n*)

The global timer vector/priority registers (GTVPRs) contain the interrupt vector and the interrupt priority as shown in Figure 10-12. They also contain the mask and activity fields.

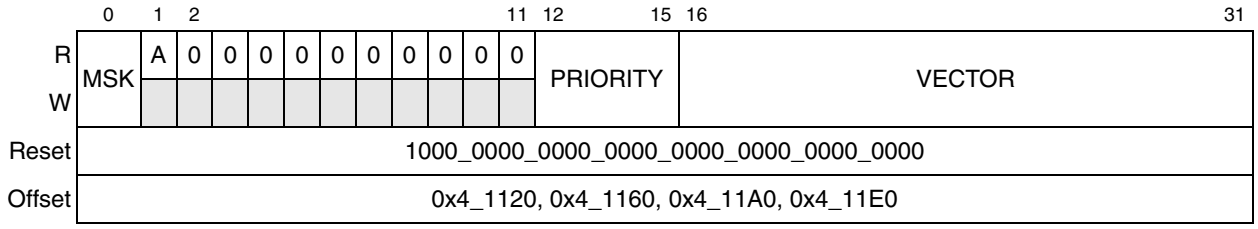


Figure 10-12. Global Timer Vector/Priority Register (GTVPR*n*)

Table 10-16 describes the GTVPR*n* fields.

Table 10-16. GTVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Inhibits interrupts from this source 0 Interrupt requests are generated when the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. This field is read-only. 0 No current interrupt activity associated with this source. 1 An interrupt has been requested by the corresponding source or is currently being serviced. The interrupt bit for this source is set in the IPR or ISR. The VECTOR and PRIORITY values should not be changed while the A bit is set.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-37.

10.3.2.5 Global Timer Destination Registers (GTDR*n*)

The global timer destination register controls the destination processor for this timer's interrupt, as shown in Figure 10-13.

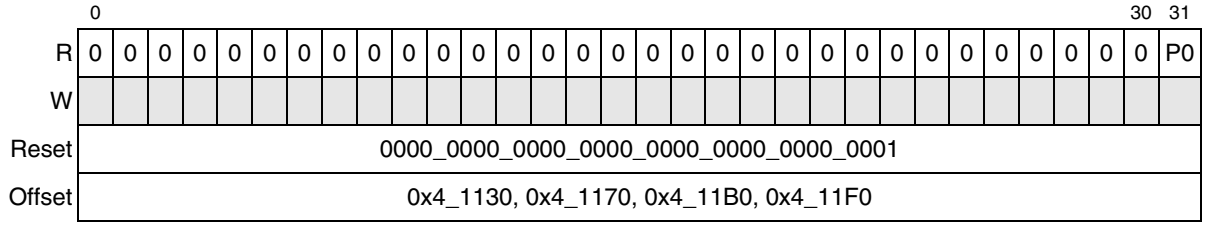


Figure 10-13. Global Timer Destination Register (GTDR*n*)

Table 10-17 describes the GTDR*n* fields.

Table 10-17. GTDR*n* Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. Because the MPC8555E is a single-core device, internally serviced interrupts are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0.

10.3.2.6 Timer Control Register (TCR)

The timer control register (TCR) shown in Figure 10-15 provides various configuration options such as count frequency and roll-over behavior.

There are two choices for the clock source for the timers: a selectable frequency ratio from the CCB clock, or the RTC signal. The TCR also provides the ability to create timers larger than the default 31-bit global timers. Timer cascade fields allow configuration of up to two 63-bit timers, one 95-bit timer or one 127-bit timer.

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field TCR[ROVR]. Setting a timer's roll-over field causes its current count register to roll over to all ones when the count reaches zero. This is equivalent to reloading the count register with 0xFFFF_FFFF instead of its base count value. Clearing a timer's associated ROVR bit ensures the timer always reloads with its base count value.

When timers are cascaded the last (most significant) counter in the cascade also affects their roll-over behavior. Cascaded timers always reload their base count when the most significant counter has decremented to zero, regardless of the settings in TCR[ROVR].

For example, timers 0, 1, and 2 can be cascaded to generate one interrupt every hour. As shown in Table 10-18, given a CCB clock of 333 MHz, letting the timer clock frequency default to 1/8 the system clock, (TCR[CLKR] = 0 sets a clock ratio of 8), provides a basic input of 41.625 MHz to timer 0. Setting timer 0 to count 41,625,000 (0x27B_25A8) timer clock cycles will generate one output per second. Setting both timers 1 and 2 to 59, and cascading all three timers, generates one interrupt every hour from timer 2.

Table 10-18. Parameters for Hourly Interrupt Timer Cascade Example

System Clock	Clock Ratio	Timer Clock	Timer 0 Count	Timer 1 Count	Timer 2 Count
333 MHz	1 / 8	41.625 MHz	41.625×10^6 (0x027B_25A8)	59[1] (0x0000_0036)	59 (0x0000_0036)

[1] Counting down from 59 through 0 requires 60 ticks.

$$(41.625 \times 106 \text{ ticks/sec}) \times (60 \text{ sec/min}) \times (60 \text{ min/hr}) = \text{Total ticks/hr generating 1 interrupt/hr}$$

Figure 10-14. Example Calculation for Cascaded Timers

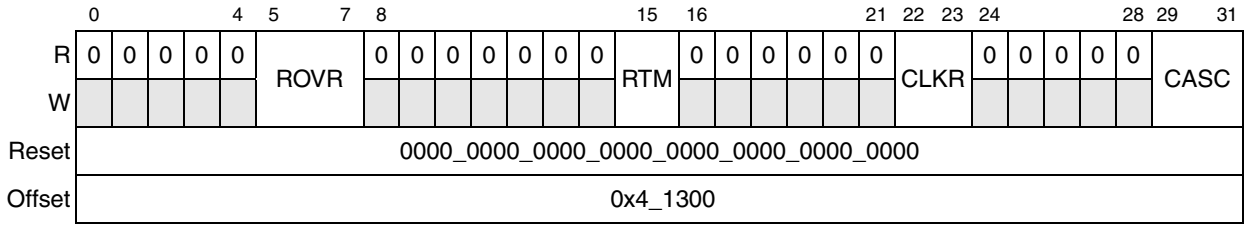


Figure 10-15. Timer Control Register (TCR)

Table 10-19 describes the TCR fields.

Table 10-19. TCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	ROVR	Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5–7 correspond to timers 2–0. Note that global timer 3 always reloads with its base count register. 0 Timer does not roll over. When the count reaches zero, current count register is reloaded with the base count register value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, current count register is reloaded with 0xFFFF_FFFF.) 000 All timers reload with base count. 001 Timers 1 and 2 reload with base count, timer 0 rolls over (reloads with 0xFFFF_FFFF) 010 Timers 0 and 2 reload with base count, timer 1 rolls over (reloads with 0xFFFF_FFFF) 011 Timer 2 reloads with base count, timers 0 and 1 roll over (reload with 0xFFFF_FFFF) 100 Timers 0 and 1 reload with base count, timer 2 rolls over (reloads with 0xFFFF_FFFF) 101 Timer 1 reloads with base count, timers 0 and 2 roll over (reload with 0xFFFF_FFFF) 110 Timer 0 reloads with base count, timers 1 and 2 roll over (reload with 0xFFFF_FFFF) 111 Timers 0, 1, and 2 roll over (reload with 0xFFFF_FFFF).
8–14	—	Reserved
15	RTM	Real time mode. Specifies the clock source for the PIC timers 0 Timer clock frequency is a ratio of the frequency of the platform (CCB) clock as determined by the CLKR field. This is the default value. 1 The RTC signal is used to clock the PIC timers. If this bit is set, the CLKR field has no meaning.
16–21	—	Reserved
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the platform (CCB) clock. The following clock ratios are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64

Table 10-19. TCR Field Descriptions (continued)

Bits	Name	Description
24–28	—	Reserved
29–31	CASC	Cascade timers. Specifies the output of particular global timers as input to others 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

10.3.3 Summary Registers

The summary registers can be read to determine which interrupt source was directed to the $\overline{\text{IRQ\_OUT}}$ output pin or caused a critical interrupt.

The $\overline{\text{IRQ\_OUT}}$ summary registers shown in Figure 10-16 and Figure 10-17 contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR[EP] is set).

The critical interrupt summary registers shown in Figure 10-18 and Figure 10-19 contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to the processor's critical interrupt signal *cint* (if the CI field in its corresponding destination register is set). The summary register bits are cleared when the corresponding interrupt that caused a bit to be setþ is negated. Note that only level sensitive interrupts can be directed to $\overline{\text{IRQ\_OUT}}$.

10.3.3.1 $\overline{\text{IRQ\_OUT}}$ Summary Register 0 (IRQSR0)

Figure 10-16 shows the IRQSR0 fields.

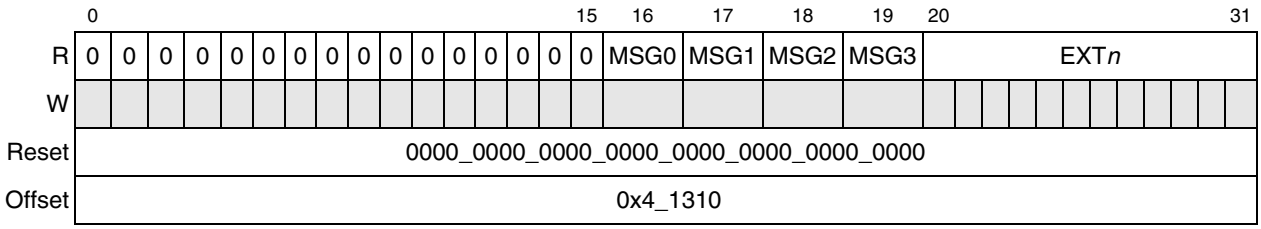


Figure 10-16. $\overline{\text{IRQ\_OUT}}$ Summary Register 0 (IRQSR0)

Table 10-20 describes the IRQSR0 fields.

Table 10-20. IRQSR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	MSG0	Message interrupt 0 status 0 Interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ 1 Interrupt is active and is directed to the $\overline{\text{IRQ\_OUT}}$ pin (that is, if the corresponding *x*IDR[EP] is set
17	MSG1	Message interrupt 1 status 0 Interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ 1 Interrupt is active and is directed to the $\overline{\text{IRQ\_OUT}}$ pin (that is, if the corresponding *x*IDR[EP] is set
18	MSG2	Message interrupt 2 status 0 Interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ 1 Interrupt is active and is directed to $\overline{\text{IRQ\_OUT}}$ (that is, if the EP corresponding *x*IDR[EP] is set
19	MSG3	Message interrupt 3 status 0 Interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$ 1 Interrupt is active and is directed to $\overline{\text{IRQ\_OUT}}$ (that is, if corresponding *x*IDR[EP] is set
20–31	EXT*n*	External interrupts 0–11. Each bit corresponds to a different interrupt according to the following: Bit Interrupt 20IRQ0 21IRQ1 . . . . 31IRQ11 0 The corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and directed to $\overline{\text{IRQ\_OUT}}$ (if the corresponding *x*IDR[EP] is set.

10.3.3.2 $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

Figure 10-17 shows the IRQSR1 fields.

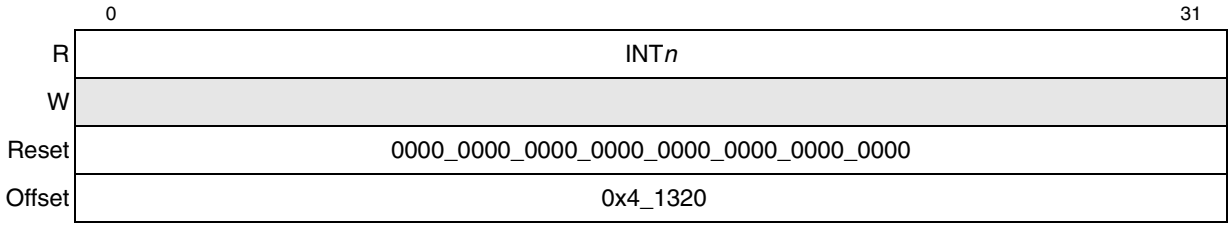


Figure 10-17. $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

Table 10-21 describes the IRQSR1 fields.

Table 10-21. IRQSR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31 status. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is directed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR[EP] is set).

10.3.3.3 Critical Interrupt Summary Register 0 (CISR0)

Figure 10-18 shows the CISR0 fields.

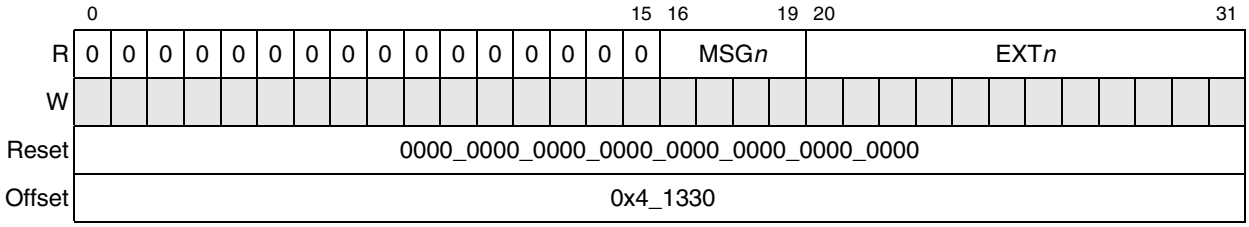


Figure 10-18. Critical Interrupt Summary Register 0 (CISR0)

Table 10-22 describes the CISR0 fields.

Table 10-22. CISR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	MSG*n*	Message interrupts 0–3. Bit 16 represents MSG0; bit 19 represents MSG3. 0 The corresponding interrupt is not active or not directed to *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (if the corresponding xIDR[CI] is set).
20–31	EXT*n*	External interrupts 0–11. Bit 20 represents IRQ0. Bit 31 represents IRQ11. 0 The corresponding interrupt is not active or not directed to *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (if the corresponding xIDR[CI] is set).

10.3.3.4 Critical Interrupt Summary Register 1 (CISR1)

Figure 10-19 shows the CISR1 fields.

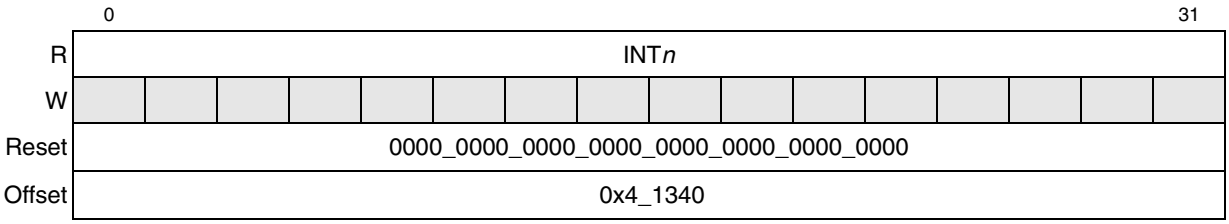


Figure 10-19. Critical Interrupt Summary Register 1 (CISR1)

Table 10-23 describes CISR1 register.

Table 10-23. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not directed to *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (if the corresponding xIDR[CI] is set).

10.3.4 Performance Monitor Mask Registers (PMMRs)

The eight performance monitor mask registers consist of four pairs of 32-bit registers, PM*n*MR0 and PM*n*MR1. Each pair can be configured to select one interrupt source (IPI, timer, message, or external) to generate a performance monitor event, The performance monitor can be configured to track this event in the performance monitor local control registers. See Section 19.3.2.2, "Performance Monitor Local Control Registers (PMLCAn, PMLCBn)."

10.3.4.1 Performance Monitor Mask Register (Lower) (PM*n*MR0)

Figure 10-20 shows the PM*n*MR0 registers. Each register is paired with a PM*n*MR1 register. Because each unreserved bit in the 64-bit pair (PM*n*MR0/1) specifies a different interrupt, only one bit in each pair can be unmasked at a time. Unmasking more than one bit per pair is considered a programming error and results in unpredictable behavior.

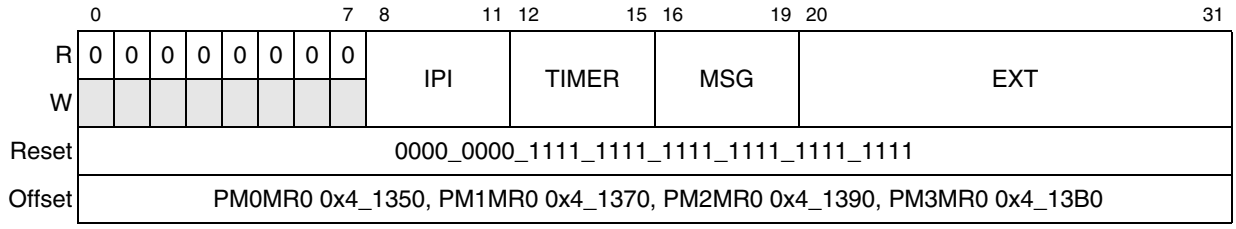


Figure 10-20. Performance Monitor Mask Registers (PM*n*MR0)

Table 10-24 describes the PM*n*MR0 fields.

Table 10-24. PM*n*MR0 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	IPI	IPI interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
12–15	TIMER	Timer interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–19	MSG	Message interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
20–31	EXT	External interrupts IRQ[0:11] 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

10.3.4.2 Performance Monitor Mask Registers (Upper) (PM*n*MR1)

Figure 10-21 shows the PM0MR1, PM1MR1, PM2MR1, and PM3MR1 fields.

	0	31
R W	INT	
Reset	1111_1111_1111_1111_1111_1111_1111_1111	
Offset	PM0MR1 0x4_1360, PM1MR1 0x4_1380, PM2MR1 0x4_13A0, PM3MR1 0x4_13C0	

Figure 10-21. Performance Monitor Mask Registers (PM*n*MR1)

Table 10-25 describes the PM*n*MR1 registers.

Table 10-25. PM*n*MR1 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

10.3.5 Message Registers

Writing to one of the four message registers (MSGR0–MSGR3) causes a messaging interrupt to the processor. Reading this register clears the messaging interrupt. Note that a messaging interrupt can also be cleared by writing a one to the corresponding status field of the PIC message status register (MSR), shown in Figure 10-24.

10.3.5.1 Message Registers (MSGR0–MSGR3)

The message registers (MSGR0–MSGR3) contain a 32-bit message.

	0	31
R W	MSG	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	MSGR0 0x4_1400, MSGR1 0x4_1410, MSGR2 0x4_1420, MSGR3 0x4_1430	

Figure 10-22. Message Registers (MSGRs)

Table 10-26 describes the MSGR registers.

Table 10-26. MSGR*n* Field Descriptions

Bits	Name	Description
0–31	MSG	Message. Contains the 32-bit message data

10.3.5.2 Message Enable Register (MER)

The message enable register (MER) shown in Figure 10-23 contains the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

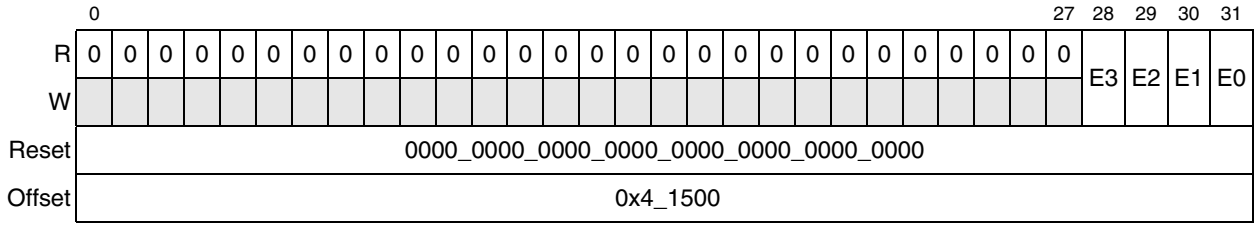


Figure 10-23. Message Enable Register (MER)

Table 10-27 describes the MER fields.

Table 10-27. MER Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	E3	Enable 3. Used to enable interrupt generation for MSGR3 0 Interrupt generation for MSGR3 disabled 1 Interrupt generation for MSGR3 enabled
29	E2	Enable 2. Used to enable interrupt generation for MSGR2 0 Interrupt generation for MSGR2 disabled 1 Interrupt generation for MSGR2 enabled
30	E1	Enable 1. Used to enable interrupt generation for MSGR1 0 Interrupt generation for MSGR1 disabled 1 Interrupt generation for MSGR1 enabled
31	E0	Enable 0. Used to enable interrupt generation for MSGR0 0 Interrupt generation for MSGR0 disabled 1 Interrupt generation for MSGR0 enabled

10.3.5.3 Message Status Register (MSR)

The PIC message status register (MSR) shown in Figure 10-24 contains status bits for each message register. The status bit is set when the corresponding messaging interrupt is active. Reading the corresponding message register or writing a 1 to a status bit or writing a 1 to a status bit clears the corresponding message interrupt and the status bit.

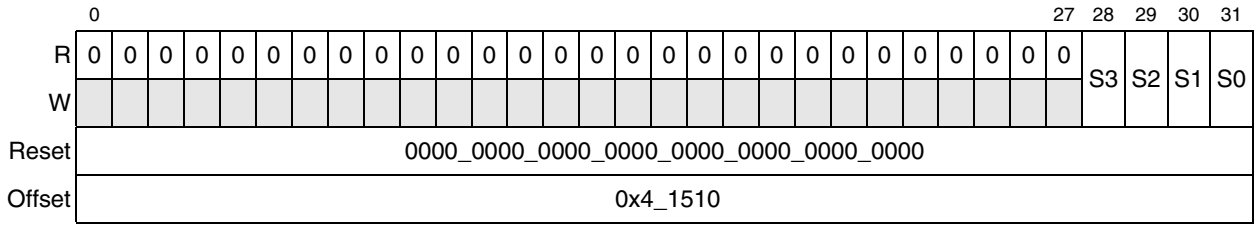


Figure 10-24. Message Status Register (MSR)

Table 10-28 describes the MSR fields.

Table 10-28. MSR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	S3	Status 3. Reports status of messaging interrupt 3. Writing a 1 clears this field. 0 Messaging interrupt 3 is not active 1 Messaging interrupt 3 is active
29	S2	Status 2. Reports status of messaging interrupt 2. Writing a 1 clears this field. 0 Messaging interrupt 2 is not active 1 Messaging interrupt 2 is active
30	S1	Status 1. Reports status of messaging interrupt 1. Writing a 1 clears this field. 0 Messaging interrupt 1 is not active 1 Messaging interrupt 1 is active
31	S0	Status 0. Reports status of messaging interrupt 0. Writing a 1 clears this field. 0 Messaging interrupt 0 is not active 1 Messaging interrupt 0 is active

10.3.6 Interrupt Source Configuration Registers

The interrupt source configuration registers control the source of each interrupt, specifying parameters such as the interrupting event, signal polarity, and relative priority.

10.3.6.1 External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

The external interrupt vector/priority registers (EIVPRs) contain polarity and sense fields for the external interrupts caused by the assertion of any of IRQ[0:11]. The format of the EIVPRs is shown in Figure 10-25.

	0	1	2					7	8	9	10	11	12	15	16	31
R	MSK	A	0	0	0	0	0	0	P	S	0	0	PRIORITY		VECTOR	
W																
Reset	1000_0000_0000_0000_0000_0000_0000_0000															
Offset	EIVPR0 0x5_0000, EIVPR1 0x5_0020, EIVPR2 0x5_0040, EIVPR3 0x5_0060, EIVPR4 0x5_0080, EIVPR5 0x5_00A0, EIVPR6 0x5_00C0, EIVPR7 0x5_00E0, EIVPR8 0x5_0100, EIVPR9 0x5_0120, EIVPR10 0x5_0140, EIVPR11 0x5_0160															

Figure 10-25. External Interrupt Vector/Priority Registers (EIVPR0—EIVPR11)

Table 10-29 describes the EIVPR fields.

Table 10-29. EIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Masks interrupts from this source 0 If the corresponding IPR bit is set an interrupt request is generated. 1 Interrupts from this source are disabled.
1	A	Activity. A read-only field indicating that an interrupt has been requested or is in-service. Note: The P (polarity), S (sense), VECTOR and PRIORITY values should not be changed while the corresponding interrupt is active, that is, while EIVPR*n* [A] is set. 0 No current interrupt activity associated with this source 1 The interrupt bit for this source is set in the IPR or ISR.
2–7	—	Reserved
8	P	Polarity. Specifies the polarity for the external interrupt. 0 Polarity is active-low or negative edge triggered 1 Polarity is active-high or positive edge-triggered
9	S	Sense. Specifies the sense for external interrupts 0 The external interrupt is edge sensitive. 1 The external interrupt is level-sensitive.
10–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. Contains the value returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in Figure 10-37.

10.3.6.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The external interrupt destination registers (EIDRs), shown in Figure 10-26, control the destination of external interrupts caused by the assertion of any of IRQ[0:11]. All external interrupts are directed to the processor 0 interrupt, *int*, by default. External interrupts can be selectively directed to $\overline{\text{IRQ\_OUT}}$ or to the processor 0 critical interrupt signal, *cint*, instead of *int* by writing to the appropriate EIDR*n* fields.

NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

	0	1	2																											30	31
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W																															
Reset	0000_0000_0000_0000_0000_0000_0000_0001																														
Offset	EIDR0 0x5_0010, EIDR1 0x5_0030, EIDR2 0x5_0050, EIDR3 0x5_0070, EIDR4 0x5_0090, EIDR5 0x5_00B0, EIDR6 0x5_00D0, EIDR7 0x5_00F0, EIDR8 0x5_0110, EIDR9 0x5_0130, EIDR10 0x5_0150, EIDR11 0x5_0170																														

Figure 10-26. External Interrupt Destination Registers (EIDRs)

Table 10-30 describes the EIDR fields. Because external interrupts can be channeled only to processor 0, the P0 bit is permanently set. As shown in Figure 10-37, if either the CI or EP bits are set, the interrupt is not sent to the processor's interrupt input.

The EP or CI fields must be set only for level-sensitive interrupts. Setting these fields for edge-sensitive interrupts does not provide reliable interrupt response.

Table 10-30. EIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External pin. Allows external interrupt to be serviced externally 0 External interrupt is serviced internally with *int* signal to the processor core 1 External interrupt is directed to $\overline{\text{IRQ\_OUT}}$ for external service
1	CI	Critical interrupt 0 External interrupt is serviced internally with *int* signal to the processor core 1 External interrupt is directed to the processor 0 as a critical interrupt with the *cint* signal
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. Because the MPC8555E is a single-core device, all interrupts that are serviced internally are always directed to processor 0. Permanently set and read only 1 Interrupt directed to processor 0

10.3.6.3 Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR31)

The internal interrupt vector/priority registers (IIVPRs), shown in Figure 10-27, have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources listed in Table 10-3. These interrupts are all level sensitive.

NOTE

Because all internal interrupts are active-high, clearing any IIVPR*n* polarity field disables that interrupt. Care should be taken to ensure this field is not inadvertently corrupted when loading or reloading IIVPRs with priority, mask, or vector data.

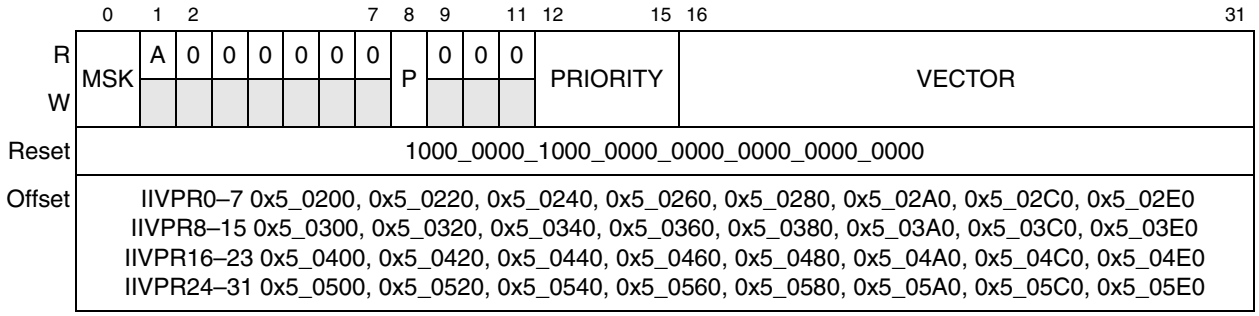


Figure 10-27. Internal Interrupt Vector/Priority Registers (IIVPRs)

Table 10-31 describes the IIVPR fields.

Table 10-31. IIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. 0 An interrupt request is generated when the corresponding IPR field is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in-service. Note this field is read only. The VECTOR and PRIORITY values should not be changed while IIVPR*n* [A] is set. 0 No current interrupt activity associated with this source 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved
8	P	Polarity. Specifies the polarity for the internal interrupt. **Note:** Because all internal interrupts are active-high, clearing this bit disables the interrupt. 0 Interrupt polarity is active-low. This value disables the interrupt. 1 Interrupt polarity is active-high. This is the reset value should not be changed.
9–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in Figure 10-37.

10.3.6.4 Internal Interrupt Destination Registers (IIDR0–IIDR31)

Internal interrupt destination registers (IIDRs), shown in Figure 10-28, have the same fields and format as EIVDRs, except that they apply to the internal interrupt sources listed in Table 10-3. Like external interrupts, internal interrupts can be directed only to processor 0.

NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

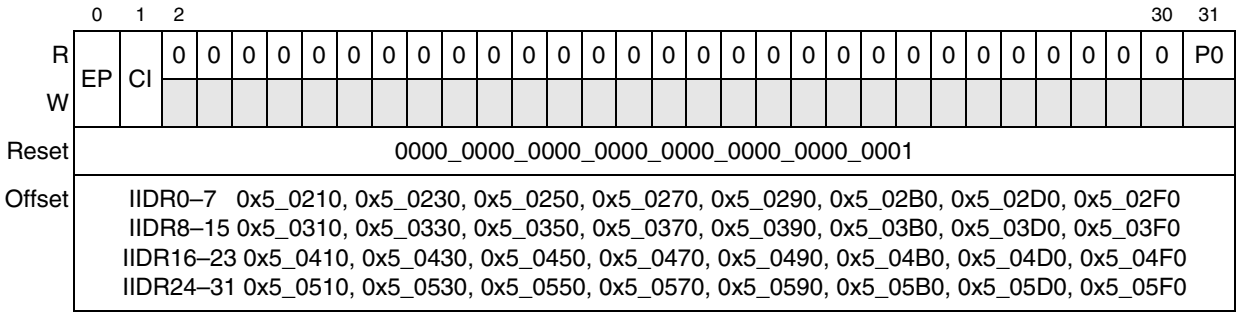


Figure 10-28. Internal Interrupt Destination Registers (IIDRs)

Table 10-32 describes the IIDR fields.

Table 10-32. IIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External pin. Allows internal interrupt to be serviced externally 0 Internal interrupt is serviced internally with *int* signal to the processor core 1 Internal interrupt is directed to $\overline{\text{IRQ\_OUT}}$ for external service. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
1	CI	Critical interrupt 0 Internal interrupt is serviced internally with *int* signal to the processor core 1 Internal interrupt is directed to the processor 0 as a critical interrupt with the *cint* signal. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. Because the MPC8555E is a single-core device, all interrupts that are serviced internally are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0

10.3.6.5 Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3)

The messaging interrupt vector/priority registers (MIVPRs) have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

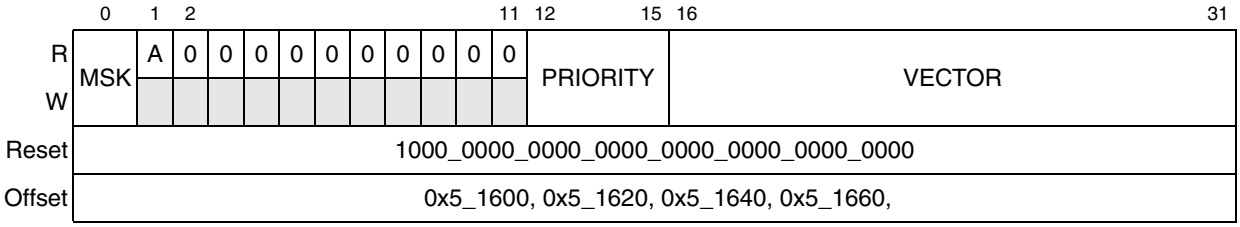


Figure 10-29. Messaging Interrupt Vector/Priority Registers (MIVPRs)

Table 10-33 describes the MIVPR fields.

Table 10-33. MIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source 0 An interrupt request is generated when the corresponding IPR field is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in-service. Note this field is read only. The VECTOR and PRIORITY values should not be changed while MIVPR*n*[A] is set. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register (IRR) shown in Figure 10-37.

10.3.6.6 Messaging Interrupt Destination Registers (MIDR0–MIDR3)

The messaging interrupt destination registers (MIDRs), shown in Figure 10-30, control the destination for the messaging interrupts. MIDR enables the user to direct the interrupt to either the external interrupt output pin ($\overline{\text{IRQ\_OUT}}$), the core's critical interrupt input (*cint*), or to its normal interrupt input (*int*).

NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

	0	1	2																												30	31
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W																																
Reset	0000_0000_0000_0000_0000_0000_0000_0001																															
Offset	0x5_1610, 0x5_1630, 0x5_1650, 0x5_1670																															

Figure 10-30. Messaging Interrupt Destination Registers (MIDRs)

Table 10-34 describes the MIDR fields.

Table 10-34. MIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External pin. Allows message interrupt to be serviced externally 0 Message interrupt is serviced internally with *int* signal to the processor core. 1 Message interrupt is directed to $\overline{\text{IRQ\_OUT}}$ for external service. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
1	CI	Critical interrupt 0 Message interrupt is serviced internally with *int* signal to the processor core. 1 Message interrupt is directed to the processor 0 as a critical interrupt with the *cint* signal. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. Because the MPC8555E is a single-core device, all interrupts that are serviced internally are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0

10.3.7 Per-CPU Registers

The OpenPIC programming model supports multiprocessor systems of up to 32 separate processors. As such, the OpenPIC interface specification provides for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. To fully comply with the OpenPIC specification, the MPC8555E PIC incorporates several of these multiprocessor capabilities. Because the value of features such as private address space for per-CPU registers and interprocessor interrupts is fully realized only in a multi-core environment, their utility in this single-core device is not intuitive.

The registers in Table 10-35 are called per-CPU registers, because they would be duplicated for each core in a multi-core device. The OpenPIC interface specifies that a copy of these registers be available to each core at the same physical address by using the ID of the processor that initiates the transaction to determine the set of per-CPU registers to access.

Table 10-35. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
IPI 0 dispatch register (IPIDR0)	0x4_0040
IPI 1 dispatch register (IPIDR1)	0x4_0050
IPI 2 dispatch register (IPIDR2)	0x4_0060
IPI 3 dispatch register (IPIDR3)	0x4_0070
Current task priority register (CTPR)	0x4_0080
Who am I register (WHOAMI)	0x4_0090
Interrupt acknowledge register (IACK)	0x4_00A0
End of interrupt register (EOI)	0x4_00B0

These addresses, shown in Table 10-35, appear in the memory map at the same offset for every processor, and are called the private access space. Because the MPC8555E has only one core, there is only one set of per-CPU registers, each register having two addresses. For example, the CTPR is located normally at 0x6_0080, and also at the private access address of 0x4_0080. While this double mapping seems superfluous on a single-core device, the purpose of this feature is to enable user code to execute correctly in a multiprocessor environment without needing to know which CPU it is running on. It is included on this device to simplify the porting of such code.

An example of how the different registers are addressed in a four-core device is illustrated in Figure 10-31. Note that when accessing a register normally, each core sources a different address. However, when accessing the same register using the per-CPU address space, each core sources the same address.

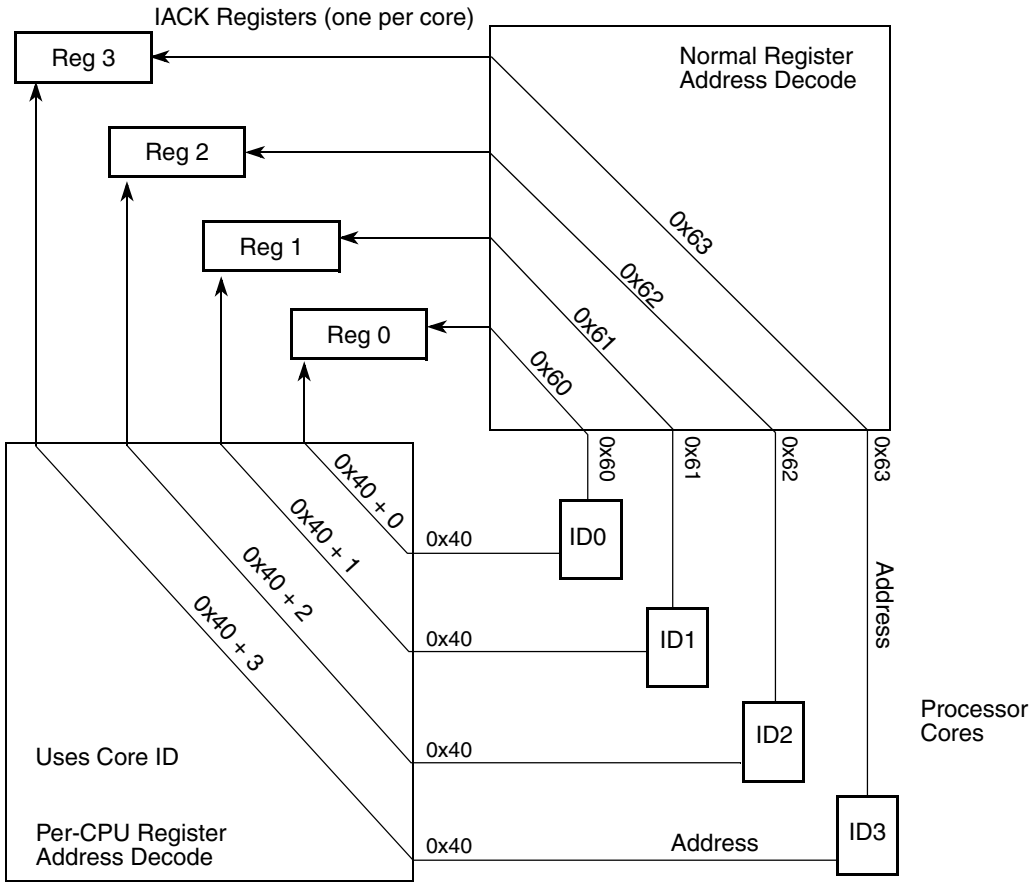


Figure 10-31. Per-CPU Register Address Decoding in a Four-Core Device

10.3.7.1 Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)

There are four interprocessor interrupt dispatch registers (IPIDRs), one for each IPI channel, as shown in Figure 10-32. Writing to an IPIDR with a bit set causes a self interrupt. While the concept of interprocessor interrupts apparently makes little sense in a single-core device, this feature can serve as a doorbell type interrupt because external bus masters can write to these registers.

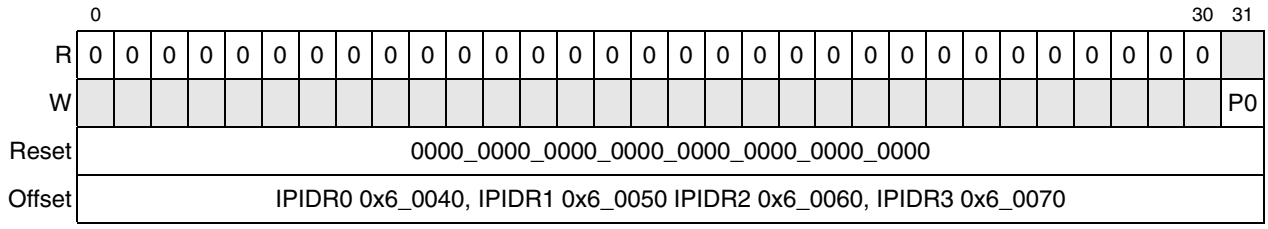


Figure 10-32. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

Table 10-36 describes the IPIDR*n* fields.

Table 10-36. IPIDR*n* Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0. Determines if processor 0 receives the interrupt (write only) 0 Processor 0 does not receive the interrupt. 1 Directs the interrupt to processor 0

10.3.7.2 Processor Current Task Priority Register (CTPR)

There is one CTPR on this device (MPC8555E) shown in Figure 10-33. Software must write the priority of the current processor task in the CTPR. The PIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to the core if the following apply:

- The interrupt is not masked.
- The priority of the interrupt is higher than the values in the CTPR and ISR.

Priority levels from 0 (lowest) to 15 (highest) are supported. Setting the task priority to 15 masks all interrupts to the processor. Hardware sets the task register to 0x0000_000F during reset or when the Px field of the processor initialization register is set.

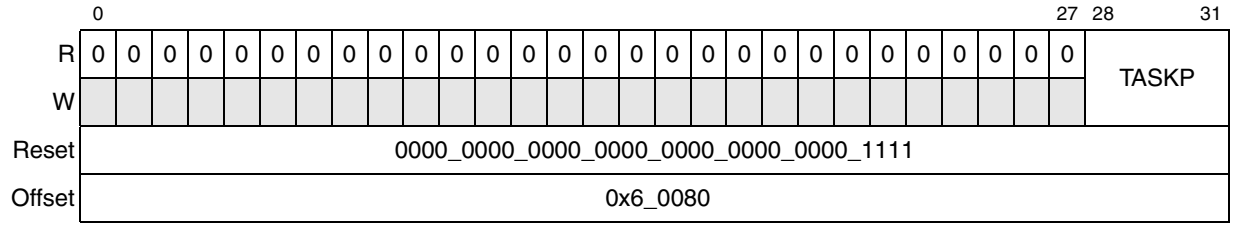


Figure 10-33. Processor Current Task Priority Register (CTPR)

Table 10-37 describes the CTPR.

Table 10-37. CTPR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TASKP	Task priority. This field is set from 0 to 15, where 15 corresponds to the highest priority for processor tasks. If CTPR[TASKP] = 0xF, no interrupts are signaled to the processor.

10.3.7.3 Who Am I Register (WHOAMI)

The processor who am I register (WHOAMI), shown in Figure 10-34, can be read by the processor to determine its physical connection to the PIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.

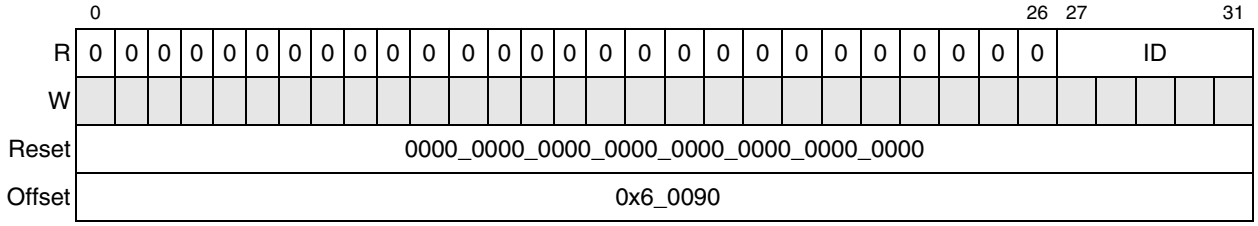


Figure 10-34. Processor Who Am I Register (WHOAMI)

Table 10-38 describes the WHOAMI fields.

Table 10-38. WHOAMI Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	ID	ID. Returns the ID of the processor reading this register. Always returns zero. 00000 Processor core

10.3.7.4 Processor Interrupt Acknowledge Register (IACK)

In systems based on processors that implement the Power Architecture technology, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK), shown in Figure 10-35. Reading the interrupt acknowledge register returns the interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the interrupt pending register is cleared for edge-sensitive interrupts.
- The in-service register (ISR) is updated.
- The interrupt signal (*int* or *cint*) to the processor is negated.

Reading this register when there is no pending interrupt returns the spurious vector value as described in Section 10.3.1.6, “Spurious Vector Register (SVR).”

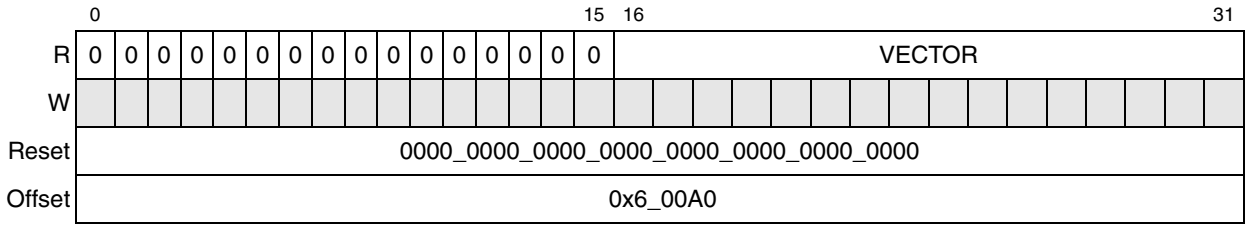


Figure 10-35. Processor Interrupt Acknowledge Register (IACK)

Table 10-39 describes the IACK fields.

Table 10-39. IACK Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only)

10.3.7.5 Processor End of Interrupt Register (EOI)

Writing to the end of interrupt (EOI) register shown in Figure 10-36 signals the end of processing for the highest-priority interrupt currently in-service by the processor. The write to the EOI updates the ISR by retiring the highest priority interrupt. Data values written to this register are ignored, and zero is assumed.

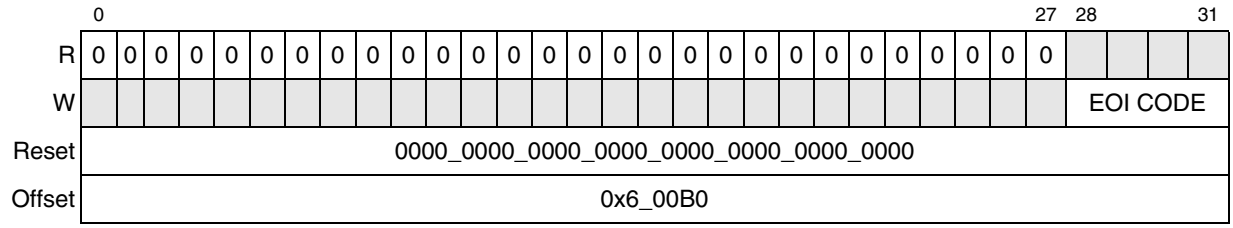


Figure 10-36. End of Interrupt Register (EOI)

Table 10-40 describes the EOI fields.

Table 10-40. EOI Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EOI CODE	0000 (write only)

10.4 Functional Description

This section is a functional description of the PIC.

10.4.1 Flow of Interrupt Control

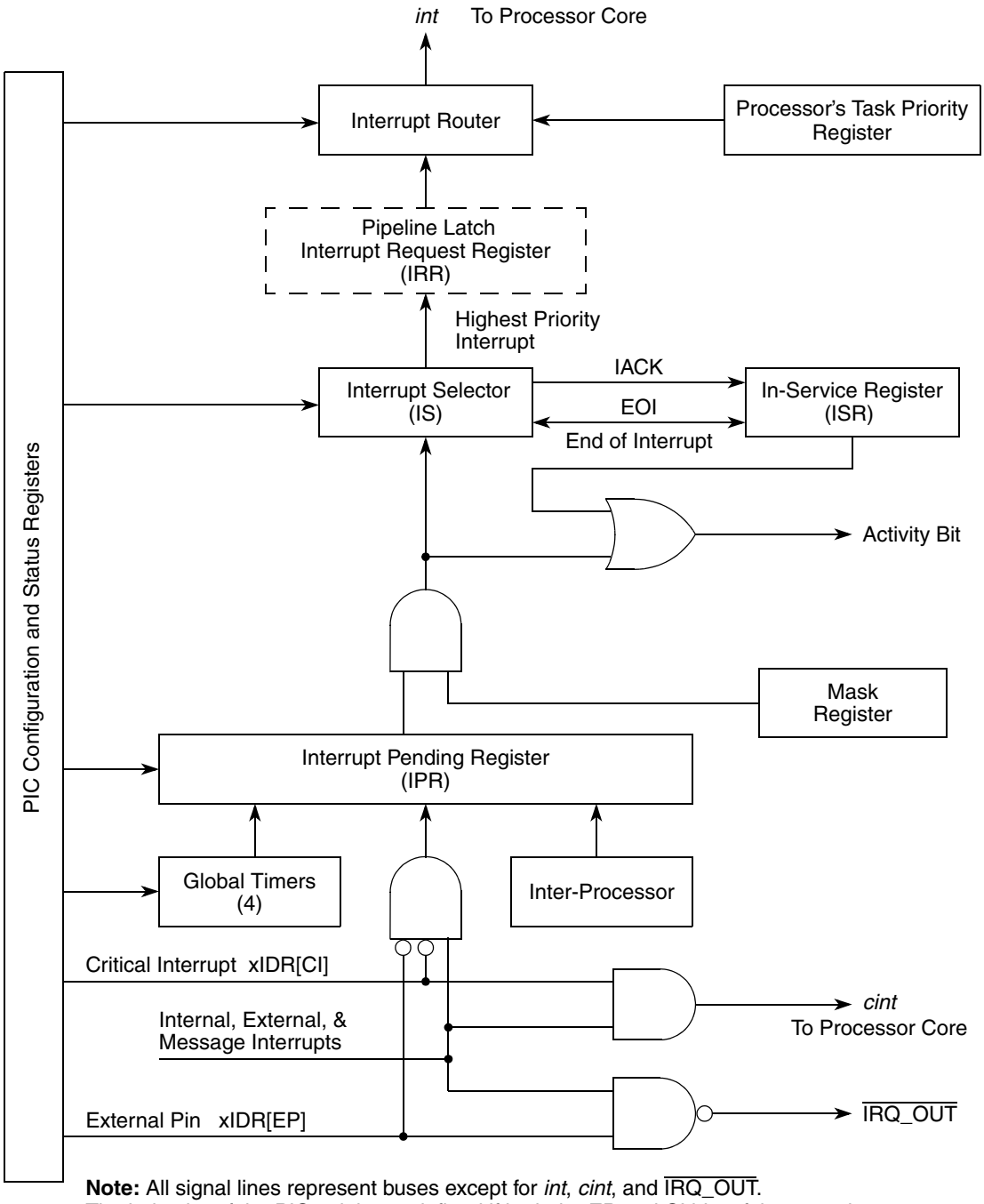
Figure 10-37 is a block diagram of the PIC unit showing the flow of interrupt processing. This figure is intended to aid in understanding and does not fully represent all internal circuitry of the actual implementation. The PIC receives interrupt signals from both external and internal sources. These signals are qualified and latched in the interrupt pending register (IPR). The IPR feeds the interrupt selector (IS). The interrupt router of the PIC monitors the outputs of its internal interrupt request register (IRR) and other configuration registers. When the priority of the interrupt latched in the IRR is higher than the value in the processor's task priority register, the interrupt router asserts the internal interrupt signal (*int*) to indicate an interrupt request to the processor. This causes the processor to vector to the external interrupt handler.

Note that the IPR, IS, and IRR are internal registers that are not accessible to the programmer.

The interrupt handler executing on the processor should then acknowledge the interrupt by explicitly reading IACK (at which point the interrupt is considered to be in-service). The PIC unit interprets this read as an interrupt acknowledge (IACK) cycle; in response, the PIC unit returns the vector associated with the interrupt source to the interrupt handler routine. The handler can further vector to different branches of interrupt handling accordingly.

Note that reading IACK also negates the interrupt signal to the processor. See Section 10.3.7.4, "Processor Interrupt Acknowledge Register (IACK)," for more details.

The internal in-service register (ISR) tracks all in-service interrupts. An interrupt is considered in-service from the time its vector is read (through an IACK cycle) until the end of interrupt (EOI) register is written, generating what the PIC considers an EOI signal.



Note: All signal lines represent buses except for *int*, *cint*, and $\overline{\text{IRQ\_OUT}}$.
The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

Figure 10-37. PIC Interrupt Processing Flow Diagram

10.4.1.1 Interrupt Source Priority

Each interrupt source is assigned a priority value through its corresponding vector/priority register. Priority values range from 0 to 15, where 15 is the highest. Interrupts are delivered only when the priority of the

interrupt source is greater than the current task priority. Therefore setting a source priority to zero inhibits that interrupt.

The PIC unit services simultaneous interrupts occurring with the same priority according to the following order:

1. MSG0–MSG3
2. IPI0–IPI3
3. timer0–timer3
4. IRQ[0:11]
5. internal0–internal31

For example, if MSG0, MSG2, and IPI0 all have the same priority and receive simultaneous interrupts, the MPC8555E services them in the following order:

1. MSG0
2. MSG2
3. IPI0

10.4.1.2 Processor Current Task Priority

The CTPR is set by system software to indicate the relative importance of the task running on the processor. The processor does not receive interrupts with a priority level equal to or lower than its current task priority. Therefore setting the current task priority to 15 for a particular processor prevents the delivery of any interrupt to the processor.

10.4.1.3 Interrupt Acknowledge

The PIC unit notifies the processor core of an interrupt by asserting the *int* signal. When the processor core acknowledges the interrupt request by reading the interrupt acknowledge register (IACK) in the PIC unit, the PIC returns the 16-bit vector associated with the interrupt source to the processor. The interrupt is then considered to be in-service, and remains in-service until the processor performs a write to the PIC unit end of interrupt register (EOI). Writing to the EOI is referred to as an EOI cycle.

10.4.2 Nesting of Interrupts

If the processor is servicing an interrupt, it can only be interrupted again if the PIC receives an interrupt request from a source with higher priority than the one currently being serviced. This is true even if software, as part of its interrupt service routine, writes a new and lower value into the CTPR.

Thus, although several interrupts may be in-service simultaneously, the code currently executing is always handling the highest priority of all the interrupts that are in service. When the processor performs an EOI cycle, this highest priority interrupt is taken out-of-service. The next EOI cycle takes the next-highest priority interrupt out-of-service, and so on. An interrupt with lower priority than those currently in-service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

10.4.3 Processor Initialization

The processor can reset itself by writing to the processor initialization register (PIR). This causes the assertion of the *core_reset* output signal. When this occurs, the processor also gets written to 0x000F to disable the delivery of any interrupts.

10.4.4 Spurious Vector Generation

Under certain circumstances, the PIC has no valid vector to return to the processor during an interrupt acknowledge cycle. In these cases, the spurious vector from the spurious vector register is returned. The following cases cause a spurious vector fetch:

- *int* is asserted in response to an externally sourced interrupt which is activated with level-sensitive logic, and the asserted level is negated before the interrupt is acknowledged.
- *int* is asserted for an interrupt source that is later masked (using the mask bit in the vector/priority register corresponding to that source) before the interrupt is acknowledged.
- *int* is asserted for an interrupt source that is later masked by an increase in the task priority level before the interrupt is acknowledged.
- An interrupt acknowledge cycle is performed by the processor in spite of the fact that the int signal has not been asserted by the PIC.

In all cases, a spurious vector is not returned if there is another pending interrupt that has sufficient priority to interrupt the processor. If such an interrupt is available, the vector for that interrupt source is returned. The EOI register should not be written in response to the spurious vector. Otherwise, a previously-accepted interrupt might be cleared unintentionally.

10.4.5 Messaging Interrupts

There are four 32-bit message registers that can be used to send 32-bit messages to the processor. A messaging interrupt is generated by writing a message register if the corresponding enable bit in the message enable/status register is set, and the interrupt is not masked. Reading the message register or writing a 1 to the status bit clears the interrupt.

10.4.6 Global Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the four internal timers of the PIC unit. The timers can be individually programmed to generate a processor interrupt when they count down to zero and can be used to generate regular periodic interrupts. Each timer has the following four configuration and control registers:

- Global timer current count register (GTCCR*n*)
- Global timer base count register (GTBCR*n*)
- Global timer vector-priority register (GTVPR*n*)
- Global timer destination register (GTDR*n*)

The timer frequency should be written to the TFRR. (All of the timers operate at this frequency.) Refer to Section 10.3.2.1, “Timer Frequency Reporting Register (TFRR),” for a description of this register.

Timer interrupts are all edge-triggered interrupts. If a timer period expires while a previous interrupt from the same source is pending or in-service, the subsequent interrupt is lost.

The timer control register (TCR) provides users with the ability to create timers larger than the 31-bit global timers. The option also exists to change the timer frequency by setting the appropriate fields of the TCR. See Section 10.3.2.6, “Timer Control Register (TCR).”

10.4.7 Reset of the PIC

The PIC unit is reset by a device power-on reset (POR) or by software that sets the GCR[RST] bit. Both of these actions cause the following:

- All pending and in-service interrupts are cleared.
- All interrupt mask bits are set.
- Polarity, sense, external pin, critical interrupt, and activity fields, are reset to their default values.
- PIR, TFRR, TCR, MER, MSR, and MSGR0–3 are cleared.
- MSG and timer destination fields are set.
- The IPI dispatch registers are cleared.
- All timer base count values are reset to zero and count inhibited.
- The CTPR[TASKP] is reset to 0xF, thus disabling interrupt delivery to the processor.
- The spurious interrupt vector resets to 0xFFFF.
- The PMMRs are reset to 0xFFFF.
- The PIC defaults to the pass-through mode (GCR[M] = 0).
- All other registers remain at their pre-reset programmed values.

The GCR[RST] bit is automatically cleared when the reset sequence is complete.

10.5 Initialization/Application Information

This section contains initialization and application information for the PIC.

10.5.1 Programming Guidelines

The following subsections contain information about programming PIC registers.

10.5.1.1 PIC Registers

Most PIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- IPI dispatch registers and the EOI register, which return zeros on reads.
- Activity bit (A) of the vector/priority registers, which returns the value according to the status of the current interrupt source.

- IACK register, which returns the vector of highest priority which is currently pending, or the spurious vector.
- Reserved fields always return 0.

The following guidelines are recommended when the PIC unit is programmed in mixed mode (GCR[M] = 1):

- All PIC registers must be located in a cache-inhibited and guarded area (through the processor MMU).
- The PIC portion of the address map must be set-up appropriately.

In addition, the following initialization sequence is recommended:

1. Write the vector, priority, and polarity values in each interrupt's vector/priority register, leaving their MSK (mask) bit set. This is required only if interrupts are used.
2. Clear the CTPR (CTPR = 0x0000_0000).
3. Program the PIC to mixed mode by setting GCR[M].
4. Clear the MSK bit in the vector/priority registers to be used.
5. Perform a software loop to clear all pending interrupts:
   — Load counter with FPR[NIRQ].
   — While counter > 0, perform IACK and EOIs to guarantee all the interrupt pending and in-service registers are cleared.
6. Set the processor CTPR value to the desired value.

Depending on the interrupt system configuration, the PIC may generate spurious interrupts to clear interrupts latched during power-up. A spurious or non-spurious vector is returned for an interrupt acknowledge cycle in this case. See the programming note below for the non-spurious case.

NOTE

Because the default polarity/sense for external interrupts is edge-sensitive, and edge-sensitive interrupts are not cleared until they are acknowledged, it is possible for the PIC to store spurious edges detected during power-up as pending external interrupts. If software permanently configures an external interrupt source to be edge-sensitive, it may receive the vector for the interrupt source and not a spurious interrupt vector when software clears the mask bit. This can occur once for any edge-sensitive interrupt when its mask bit is first cleared and the PIC is in mixed mode.

To avoid having to handle a false interrupt for this case, software can clear the PIC interrupt pending register of these spurious edge detections by first configuring the polarity/sense of external interrupt sources to be level-sensitive; high-level if the input is a positive-edge source, low-level if it's a negative-edge source (while the mask bit remains set). After this is complete, configuring the external interrupt source as edge-sensitive will not cause a false interrupt.

10.5.1.2 Changing Interrupt Source Configuration

To change the vector, priority, polarity, sense or destination of an active (unmasked) interrupt source, the following sequence should be performed:

1. Mask the source using the mask (MSK) bit in the vector/priority register.
2. Wait for the activity (A) bit for that source to be cleared.
3. Make the desired changes.
4. Unmask the source.

Chapter 11
I²C Interface

This chapter describes the inter-IC (IIC or I²C) bus interface implemented on this device.

11.1 Introduction

The 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 11-1 shows a block diagram of the I²C interface.

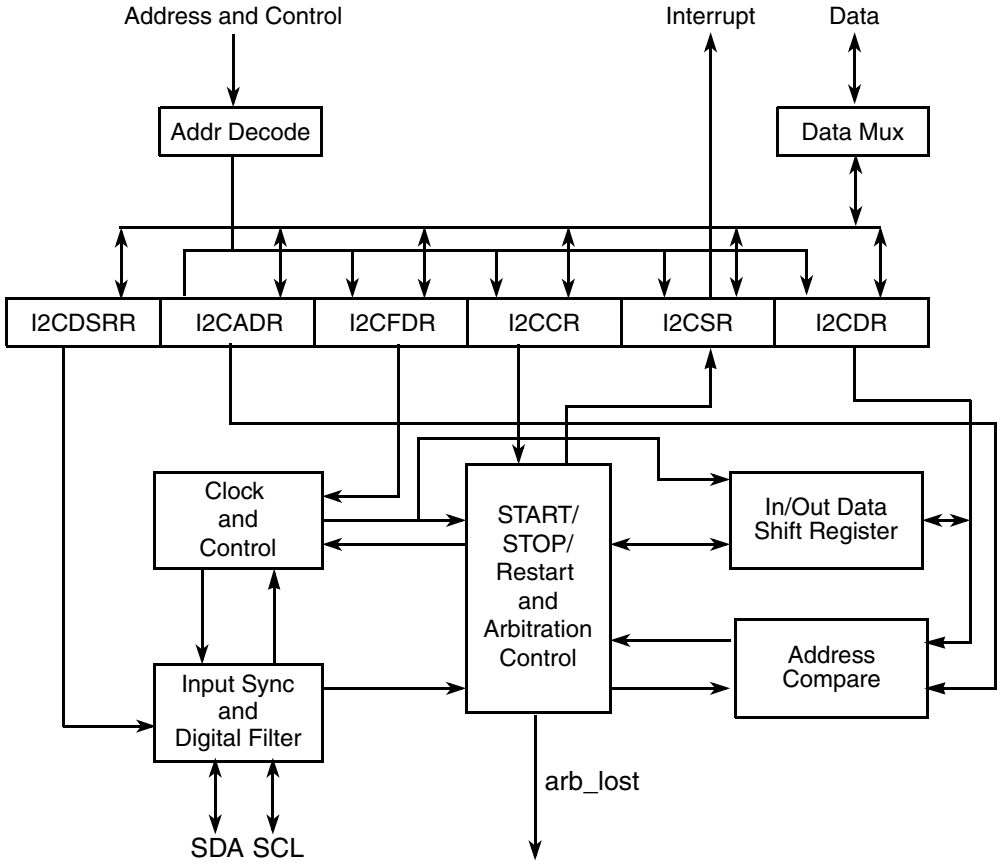


Figure 11-1. I²C Block Diagram

11.1.1 Overview

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.

11.1.2 Features

The I²C interface includes the following features:

- Two-wire interface
- Multiple-master operation
- 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

11.1.3 Modes of Operation

The I²C unit on this device can operate in one of the following modes:

- Master mode—The I²C is the driver of the SDA line. 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 not the driver of the SDA line. The module must be enabled before a START condition from a non-I²C master is detected.
- Interrupt-driven byte-to-byte data transfer—When successful slave addressing is achieved (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ 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—This mode can be used to initialize the configuration registers in the device after the I²C module is initialized. Note that the device powers up with boot sequencer mode disabled as a default, but this mode can be selected with the cfg_boot_seq[0:1] power-on reset (POR) configuration signals that are located on the LGPL3 and LGPL5 signals.

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.

11.2 External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

11.2.1 Signal Overview

The I²C interface uses the SDA and SCL signals, described in Table 11-1, for data transfer. Note that the signal patterns driven on SDA represent address, data, or read/write information at different stages of the protocol.

Table 11-1. I²C Interface Signal Descriptions

Signal Name	Idle State	I/O	State Meaning
Serial Clock (IIC_SCL)	HIGH	I	When the I²C module is idle or acts as a slave, SCL defaults as an input. The unit uses SCL to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.
		O	As a master, the I²C module drives SCL along with SDA when transmitting. As a slave, the I²C module drives SCL low for data pacing.
Serial Data (IIC_SDA)	HIGH	I	When the I²C module is idle or in a receiving mode, SDA defaults as an input. The unit receives data from other I²C devices on SDA. The bus is assumed to be busy when SDA is detected low.
		O	When writing as a master or slave, the I²C module drives data on SDA synchronous to SCL.

11.2.2 Detailed Signal Descriptions

SDA and SCL, described in Table 11-2, serve as a communication interconnect with other devices. All devices connected to these two 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 *MPC8555E PowerQUICC™ III Integrated Processor Family Hardware Specifications* for the electrical characteristics of these signals.

Table 11-2. I²C Interface Signal—Detailed Signal Descriptions

Signal	I/O	Description	
IIC_SCL	I/O	Serial clock. Performs as an input when the device is programmed as an I²C slave. SCL also performs as an output when the device is programmed as an I²C master.	
	O	As outputs for the bidirectional serial clock, these signals operate as described below.	
		State Meaning	Asserted/Negated—Driven along with SDA as the clock for the data.
	I	As inputs for the bidirectional serial clock, these signals operate as described below.	
		State Meaning	Asserted/Negated—The I²C unit uses this signal to synchronize incoming data on SDA. The bus is assumed to be busy when this signal is detected low.
IIC_SDA	I/O	Serial data. Performs as an input when the device is in a receiving mode. SDA also performs as an output signal when the device is transmitting (as an I²C master or a slave).	
	O	As outputs for the bidirectional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated— Data is driven.
	I	As inputs for the bidirectional 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 is detected low.

11.3 Memory Map/Register Definition

Table 11-3 lists the I²C-specific registers and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 11-3.

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.

All I²C registers are one byte wide. Reads and writes to these registers must be byte-wide operations.

Table 11-3. I²C Memory Map

Offset	I²C Register	Access	Reset	Section/Page
I²C Registers				
Block Base Address: 0x0_3000				
0x000	I2CADR—I²C address register	R/W	0x00	11.3.1.1/11-5
0x004	I2CFDR—I²C frequency divider register	R/W	0x00	11.3.1.2/11-6
0x008	I2CCR—I²C control register	Mixed	0x00	11.3.1.3/11-7
0x00C	I2CSR—I²C status register	Mixed	0x81	11.3.1.4/11-9
0x010	I2CDR—I²C data register	R/W	0x00	11.3.1.5/11-10
0x014	I2CDFSRR—I²C digital filter sampling rate register	R/W	0x10	11.3.1.6/11-11

11.3.1 Register Descriptions

This section describes the I²C registers in detail.

NOTE

Reserved bits should always be written with the value they returned 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 note does not apply to the I²C data register (I2CDR).

11.3.1.1 I²C Address Register (I2CADR)

Figure 11-2 shows the I2CADR 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.

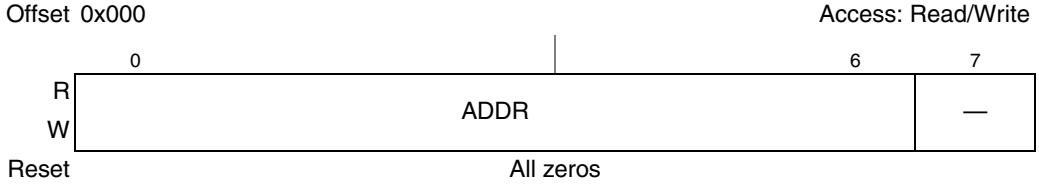


Figure 11-2. I²C Address Register (I2CADR)

Table 11-4 describes the fields of I2CADR.

Table 11-4. I2CADR Field Descriptions

Bits	Name	Description
0–6	ADDR	Slave address. Contains the specific slave address that is used by the I²C interface. Note that the default mode of the I²C interface is slave mode for an address match. Note that an address match is one of the conditions that can cause I2CSR[MIF] to be set, signaling an interrupt pending condition.
7	—	Reserved

11.3.1.2 I²C Frequency Divider Register (I2CFDR)

Figure 11-3 shows the bits of the I²C frequency divider register. Refer to application note AN2919, *Determining the I2C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

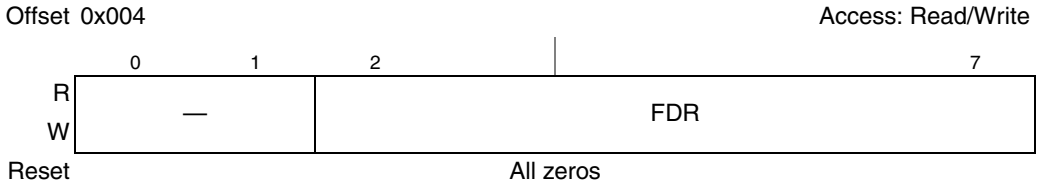


Figure 11-3. I²C Frequency Divider Register (I2CFDR)

Table 11-5 describes the bit settings of I2CFDR. It also maps the I2CFDR[FDR] field to the clock divider values.

Table 11-5. I2CFDR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	FDR	Frequency divider ratio. Used to prescale the clock for bit rate selection. The serial bit clock frequency of SCL is equal to the CCB clock divided by the designated divider. Note that the frequency divider value can be changed at any point in a program. The serial bit clock frequency divider selections are described as follows:

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				

11.3.1.3 I²C Control Register (I2CCR)

Figure 11-4 shows the I²C control register, I2CCR.

Offset 0x008 Access: Mixed

	0	1	2	3	4	5	6	7
R	MEN	MIEN	MSTA	MTX	TXAK		—	BCST
W						RSTA		
Reset	All zeros							

Figure 11-4. I²C Control Register (I2CCR)

Table 11-6 describes the bit settings of the I2CCR.

Table 11-6. I2CCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. This bit controls the software reset of the I²C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other control register bits have any effect. All I²C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I²C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I²C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
2	MSTA	Master/slave mode START 0 When this bit is changed from one to zero, a STOP condition is generated and the mode changes from master to slave. 1 Cleared without generating a STOP condition when the master loses arbitration. When this bit is changed from zero to one, a START condition is generated on the bus, and master mode is selected.
3	MTX	Transmit/receive mode select. This bit selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2CSR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. The MTX bit is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. This bit specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit only applies when the I²C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA) is sent out to the bus at the 9th clock after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
5	RSTA	Repeated START. 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. Note that this bit is not readable, which means if a read is performed to I2CCR[RSTA], a zero value will be returned. 0 No START condition is generated 1 Generates repeated START condition
6	—	Reserved
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I²C to accept broadcast messages at address zero

11.3.1.4 I²C Status Register (I2CSR)

The I²C status register, shown in Figure 11-5, is read only with the exception of the MIF and MAL bits, which can be cleared by software. The MCF and RXAK bits are set at reset; all other I2CSR bits are cleared on reset.

Offset 0x00C Access: Mixed

	0	1	2	3	4	5	6	7
R	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
W								
Reset	1	0	0	0	0	0	0	1

Figure 11-5. I²C Status Register (I2CSR)

Table 11-7 describes the bit settings of the I2CSR.

Table 11-7. I2CSR 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: • When I2CSR is read in receive mode or • When I2CDR is written in transmit mode 1 Byte transfer is completed
1	MAAS	Addressed as a slave. When the value in I2CDR matches with the calling address, this bit is set. The processor is interrupted, if I2CCR[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit. 0 Not addressed as 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 I²C bus is idle 1 I²C 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 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. This will also be set if this I²C drives an address of all 0s and broadcast mode is enabled.
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: • 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 this bit, the processor can select slave transmit/receive mode according to the command of the master.

Table 11-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIEN] 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: • One byte of data is transferred (set at the falling edge of the 9th clock). • The value in I2CADR matches with the calling address in slave-receive mode. • Arbitration is lost.
7	RXAK	Received acknowledge. The value of SDA 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

11.3.1.5 I²C Data Register (I2CDR)

The I2C data register is shown in Figure 11-6.

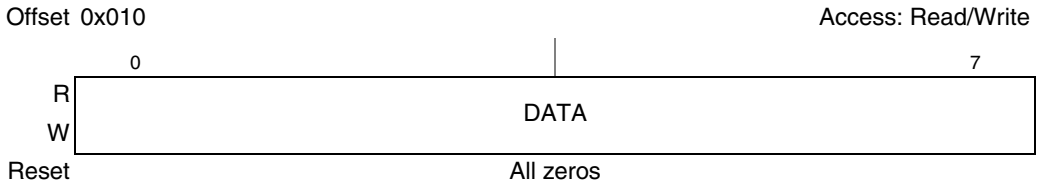


Figure 11-6. I²C Data Register (I2CDR)

Table 11-8 shows the bit descriptions for I2CDR.

Table 11-8. I2CDR 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 I²C interface performs as the master. A data transfer is initiated when data is written to the I2CDR. 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 I²C module to receive the next byte of data on the I²C interface. In slave mode, the same function is available after it is addressed. Note that the very first read is always a dummy read.

11.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in Figure 11-7. Refer to application note AN2919, "Determining the I2C Frequency Divider Ratio for SCL," for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

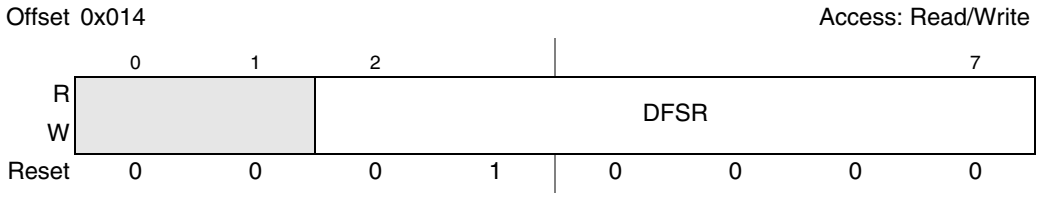


Figure 11-7. I²C Digital Filter Sampling Rate Register (I2CDFSRR)

Table 11-9 shows the field descriptions for I2CDFSRR.

Table 11-9. I2CDFSRR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	DFSR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I²C bus. The resulting sampling rate is calculated by dividing the platform (CCB clock) frequency by the non-zero value of DFSR.

11.4 Functional Description

The I²C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. After the boot sequencer has completed (when powered up in boot sequencer mode), the I²C interface will perform as a slave receiver.

11.4.1 Transaction Protocol

A standard I²C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 11-8 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 sections.

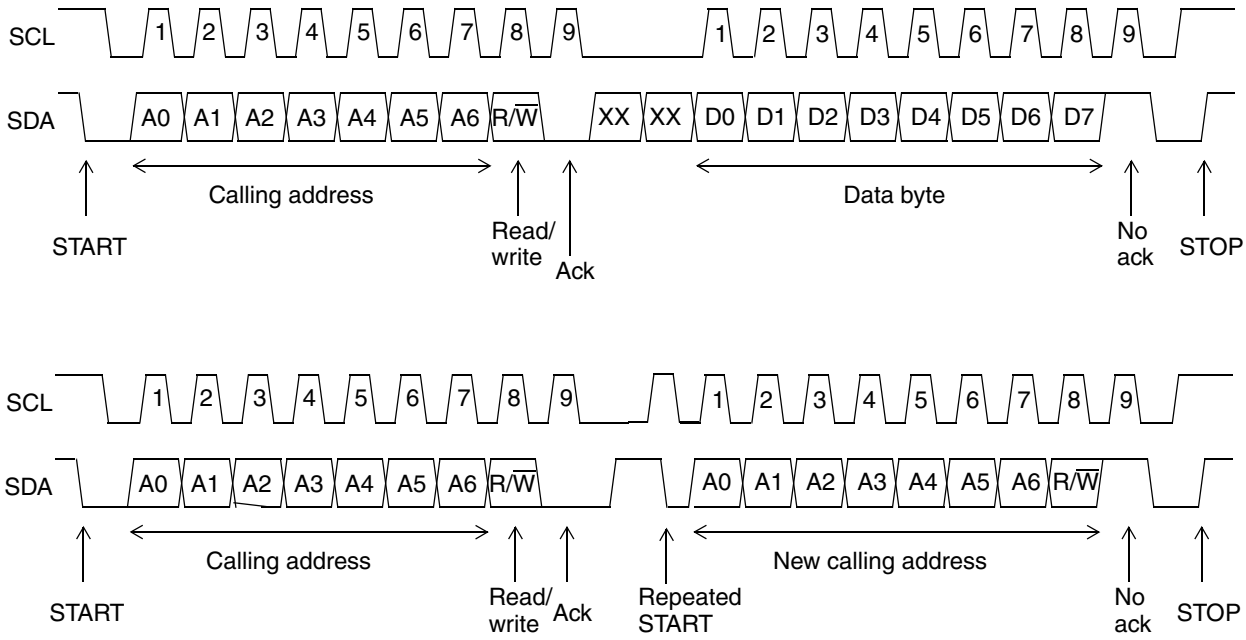


Figure 11-8. I²C Interface Transaction Protocol

11.4.1.1 START Condition

When the I²C bus is not engaged (both SDA and SCL lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 11-8, a START condition is defined as a high-to-low transition of SDA while SCL is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all slaves. The START condition is initiated by a software write that sets I2CCR[MSTA].

11.4.1.2 Slave Address Transmission

The first byte of data is transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/$\overline{W}$ bit, which indicates the direction of the data being transferred to the slave. Each slave in the system has a unique address. In addition, 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; if this is attempted, the results are boundedly undefined.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (pulling the SDA signal low at the 9th clock) as shown in Figure 11-8. 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 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.

The I²C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero; however the I²C module will not check the R/$\overline{W}$ 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 I2CDR with I2CCR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

Each data byte is 8 bits long. Data bits can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 11-8. There is one clock pulse on SCL for each data bit, and the most significant bit (msb) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device by pulling the SDA line low at the 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 line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

11.4.1.3 Repeated START Condition

Figure 11-8 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.

11.4.1.4 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 signal while SCL is high. For more information, see Figure 11-8. 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 I2CCR[MSTA].

As described in Section 11.4.1.3, “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.

11.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in this I²C module.

11.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I²C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when an SDA rise occurs while SCL is high.

- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

11.4.1.5.2 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 output is pulled low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless it is performing a START, STOP, or restart condition. Otherwise, the SDA output is held constant.

The SDA signal is pulled low when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Data bit (transmit)
  - Ack bit (receive)
  - START condition
  - STOP condition
  - Restart condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - Ack bit (receive)

The SCL signal corresponds to the internal SCL signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Bus owner
  - Lost arbitration
  - START condition
  - STOP condition
  - Restart condition begin
  - Restart condition end
- Slave mode
  - Address cycle
  - Transmit cycle
  - Ack cycle

11.4.1.6 Address Compare—Implementation Details

Address compare block determines if a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The three performed address comparisons are described as follows:

- Whether a broadcast message has been received, to update the I2CSR
- Whether the module has been addressed as a slave, to update the I2CSR and to generate an interrupt
- If the address transmitted by the current master matches the general broadcast address

11.4.2 Arbitration Procedure

The I²C interface is a true multiple-master bus that allows more than one master device to be connected on it. 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 while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I²C unit sets the I2CSR[MAL] status bit 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 results in the current bus master of the I²C interface losing arbitration, after which bus operations return to normal.

11.4.2.1 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 line while attempting to drive a 1, tries to generate a START or restart at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and I2CSR[MAL] is set) under the following conditions:

- SDA samples low when the master drives high during an address or data-transmit cycle (transmit).
- SDA 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 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.

11.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

11.4.4 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 restart condition
  - — Transmit data
  - — Receive data
- Slave mode
  - — Transmit data
  - — Receive data
  - — Receive slave address after START or restart condition

11.4.4.1 Clock Synchronization

Due to the wire AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

11.4.4.2 Input Synchronization and Digital Filter

The following sections describes the synchronizing of the input signals, and the filtering of the SCL and SDA lines in detail.

11.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

11.4.4.2.2 Filtering of SCL and SDA Lines

The SCL and SDA inputs are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

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 frequency register to control the filtered sampling rate.

11.4.4.3 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven the SCL line low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period, then the resulting SCL bus signal low period is stretched.

11.4.5 Boot Sequencer Mode

If boot sequencer mode is selected on POR (by the settings on the LGPL3 and LGPL5 reset configuration signals, as described in Section 4.4.3.6, “Boot Sequencer Configuration”), the I²C module communicates with one or more EEPROMs through the I²C interface. The boot sequencer accesses the I²C serial ROM device at the interface frequency designated by the default value of the I2CDFR[DFR] field, 0x2C, which corresponds to a divider of 1280. See Section 11.3.1.2, “I²C Frequency Divider Register (I2CFDR),” for additional details of the I2CFDR. The EEPROM(s) can be programmed to initialize one or more configuration registers of this integrated device.

The 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, and this extended addressing mode is selectable during POR with a different encoding on the LGPL3 and LGPL5 reset configuration signals (see Section 4.4.3.6, “Boot Sequencer Configuration”). In this mode, only one EEPROM device may 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 device to hang and the $\overline{\text{HRESET\_REQ}}$ signal to assert externally. 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 11.4.5.1, “EEPROM Calling Address.” There should be no other I²C traffic when the boot sequencer is active.

Note that as described in Section 4.4.3.6, “Boot Sequencer Configuration,” the default value for the LGPL3 and LGPL5 reset configuration pins is 0b11, which corresponds to the I²C boot sequencer being disabled at power-up.

11.4.5.1 EEPROM Calling Address

The MPC8555E uses 0b101_0000 for the EEPROM calling address. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. If more EEPROMs are used, they are addressed in sequential order.

11.4.5.2 EEPROM Data Format

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 a series of configuration registers (known as register preloads) programmed into the EEPROM. Each configuration register should be programmed according to a particular format, as shown in Figure 11-9. The first 3 bytes hold the attributes and address offset, as follows. The attributes contained are alternate configuration space (ACS), byte enables, and continue (CONT). The boot sequencer expects the address offset to be a 32-bit (word) offset, that is, the 2 low-order bits are not included in the boot sequencer command. For example, to access LAWBAR0 (byte offset of 0x00C08), the boot sequencer ADDR[0:17] should be set to 0x00302.

After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of the transaction. Byte enables should be asserted for any byte that will be written to the configuration register, 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 LSB of data (data[24:31]).

By setting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer. Otherwise, CCSRBAR is prepended to the EEPROM address.

If CONT is cleared, the first 3 bytes, including ACS, the byte enables, and the address, must also be cleared. Also, the data contains the final cyclic redundancy check (CRC). A CRC-32 algorithm is used to check the integrity of the data. The polynomial used is:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

CRC values are calculated using the above polynomial with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000. The CRC should cover all bytes stored in the EEPROM prior to 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). If a preamble or CRC fail is detected, the device hangs and the external $\overline{\text{HRESET\_REQ}}$ signal asserts. If there is a preamble fail, the boot sequencer may continue to pull I²C pins low until a hard reset occurs.

0	1 – 4	5	6 – 7
ACS	BYTE_EN	CONT	ADDR[0–1]
ADDR[2–9]			
ADDR[10–17]			
DATA[0–7]			
DATA[8–15]			
DATA[16–23]			
DATA[24–31]			

Figure 11-9. EEPROM Data Format for One Register Preload Command

Figure 11-10 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[0–1]		First Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
ACS	BYTE_EN				1	ADDR[0–1]		Second Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
. . .								

Figure 11-10. EEPROM Contents

ACS	BYTE_EN				1	ADDR[0–1]		Last Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
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 11-10. EEPROM Contents (continued)

11.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. Figure 11-11 is a recommended flowchart for I²C interrupt service routines.

The I²C registers in this chapter are shown in big-endian format. If the system is in little-endian mode, software must swap the bytes appropriately. This appropriate byte swapping is needed as I²C registers are byte registers. Also, an **msync** assembly instruction must be executed after each I²C register read/write access to guarantee in-order execution.

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 status bits returned after an interrupt are not consistent with what was expected due to illegal I²C bus protocol behavior.

11.5.1 Initialization Sequence

A hard reset initializes all 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 I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the CCB (platform) clock. Note that the platform frequency must first be divided by two; see Section 11.3.1.2, "I²C Frequency Divider Register (I2CFDR)," for more details.
3. Update I2CADR to define the slave address for this device.

4. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2CCR[MEN] to enable the I²C interface.

11.5.2 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, test the state of I2CSR[MBB] to check whether the serial bus is free (I2CSR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.
3. Write the slave address being called into I2CDR. The data written to I2CDR[0–6] comprises the slave calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

The scenario above assumes that the I²C interrupt bit (I2CSR[MIF]) is cleared. If MIF is set at any time, an I²C interrupt is generated (provided interrupt reporting is enabled with I2CCR[MIEN] =1) so that the I²C interrupt handler can handle the interrupt.

11.5.3 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. The I²C interrupt bit (I2CSR[MIF]) is also set and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set). In the interrupt handler, software must take the following steps:

1. Clear I2CSR[MIF]
2. Read the contents of the I²C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this causes I2CSR[MCF] to be cleared. See Section 11.5.8, “Interrupt Service Routine Flowchart.”

When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CCR[MTX] must be toggled at this stage. See Section 11.5.8, “Interrupt Service Routine Flowchart.”

If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] must be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I²C signals have time to settle. Thus, when polling I2CSR[MIF] (or any other I2CSR bits), software delays may be needed in order to give the I²C signals sufficient time to settle.

During slave-mode address cycles (I2CSR[MAAS] is set), I2CSR[SRW] should be read to determine the direction of the subsequent transfer and I2CCR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2CSR[SRW] is not valid and I2CCR[MTX] must be read to determine the direction of the current transfer. See Section 11.5.8, “Interrupt Service Routine Flowchart,” for more details.

11.5.4 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 (I2CCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has already been transferred on the I²C interface, so the last byte will not receive the data acknowledge (because I2CCR[TXAK] is set). For 1-byte transfers, a dummy read should be performed by the interrupt service routine (see Section 11.5.8, "Interrupt Service Routine Flowchart"). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I²C controller automatically generates a STOP if I2CCR[TXAK] is set. Therefore, I2CCR[TXAK] must be set before allowing the I²C module to receive the last data byte on the I²C bus. Eventually, I2CCR[TXAK] needs to be cleared again for subsequent I²C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

11.5.5 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 I2CCR[RSTA].

11.5.6 Generation of SCL When SDA Low

It is sometimes necessary to force the I²C module to become the I²C bus master out of reset and drive SCL (even though SDA may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I²C devices to be reset. Thus, SDA can be driven 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 so that the device driving SDA can finish its transaction:

1. Disable the I²C module and set the master bit by setting I2CCR to 0x20
2. Enable the I²C module by setting I2CCR to 0xA0
3. Read the I2CDR
4. Return the I²C module to slave mode by setting I2CCR to 0x80

11.5.7 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 I2CSR[MAAS] is set, software should set the transmit/receive mode select bit (I2CCR[MTX]) according to the R/$\overline{W}$ command bit (I2CSR[SRW]). Writing to I2CCR clears 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 I2CDR for slave transmits or dummy reading from I2CDR in slave-receive mode. The slave drives SCL low between byte transfers. SCL is released when the I2CDR is accessed in the required mode.

11.5.7.1 Slave Transmitter and Received Acknowledge

In the slave transmitter routine, the received acknowledge bit (I2CSR[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 (I2CSR[RXAK] is set), the slave transmitter interrupt routine must clear I2CCR[MTX] to switch the slave from transmitter to receiver mode. A dummy read of I2CDR then releases SCL so that the master can generate a STOP condition. See Section 11.5.8, "Interrupt Service Routine Flowchart."

11.5.7.2 Loss of Arbitration and Forcing of Slave Mode

When a master loses arbitration the following conditions all occur:

- I2CSR[MAL] is set
- I2CCR[MSTA] is cleared (changing the master to slave mode)
- An interrupt occurs (if enabled) at the falling edge of the 9th clock of this transfer

Thus, the slave interrupt service routine should first test I2CSR[MAL] and software should clear it if it is set. See Section 11.4.2.1, "Arbitration Control," for more information.

11.5.8 Interrupt Service Routine Flowchart

Figure 11-11 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 I2CCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that an **msync** instruction follow each I²C register read or write to guarantee in-order instruction execution.

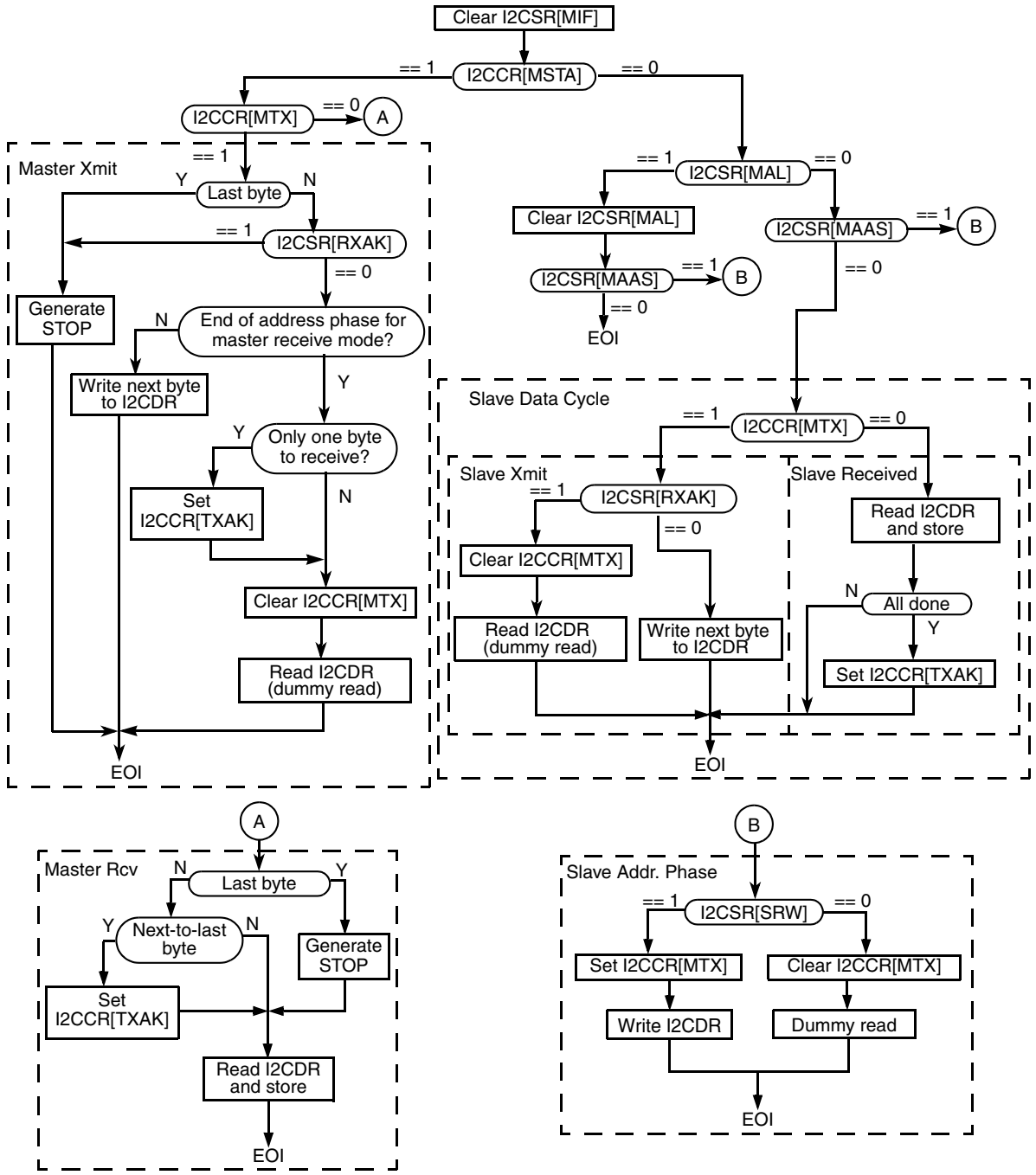


Figure 11-11. Example I²C Interrupt Service Routine Flowchart

Chapter 12
DUART

This chapter describes the dual universal asynchronous receiver/transmitters (DUART). It describes the functional operation, the DUART initialization sequence, and the programming details for the DUART registers and features.

12.1 Overview

The DUART consists of two 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 core complex bus (CCB) 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 12-1, 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

12.1.1 Features

The DUART includes these distinctive 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 CCB clock by 1 to ($2^{16} - 1$) and generate a 16x clock for the transmitter and receiver engines

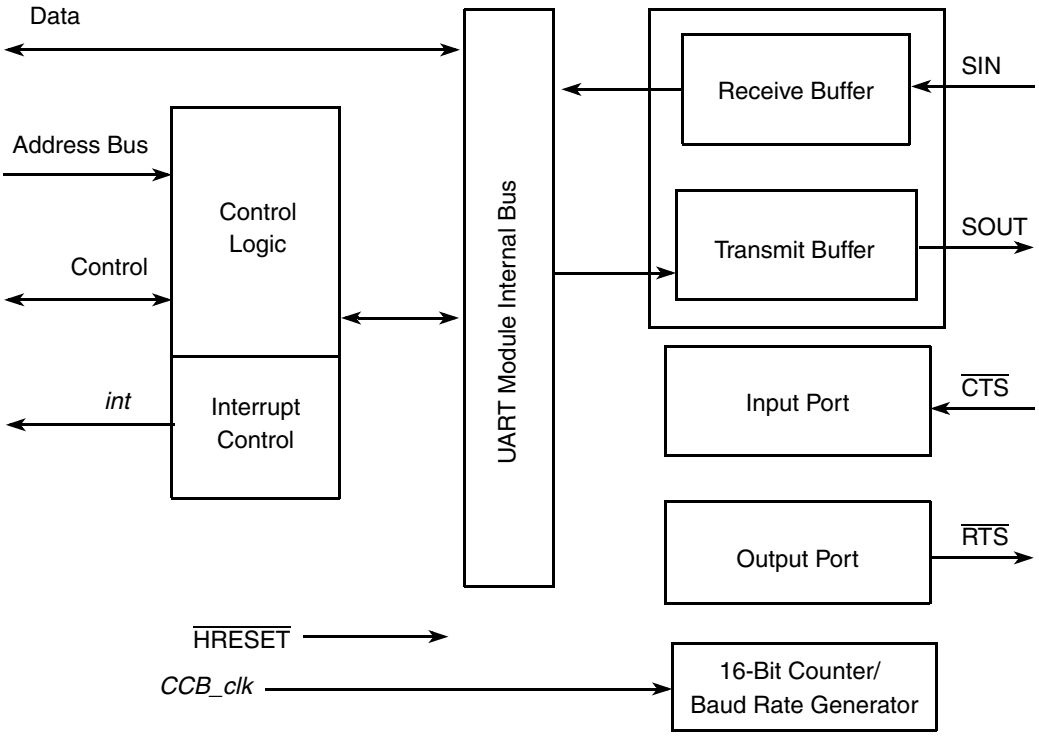


Figure 12-1. UART Block Diagram

- 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 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

12.1.2 Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB 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.

12.2 External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

12.2.1 Signal Overview

Table 12-1 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 12-1. DUART Signal Overview

Signal Name	I/O	Pins	Reset Value	State Meaning
UART_SIN[0:1]	I	2	1	Serial in data UART0 and UART1
UART_SOUT[0:1]	O	2	1	Serial out data UART0 and UART1
$\overline{\text{UART\_CTS}}$[0:1]	I	2	1	Clear to send UART0 and UART1
$\overline{\text{UART\_RTS}}$[0:1]	O	2	1	Request to send UART0 and UART1

12.2.2 Detailed Signal Descriptions

The DUART signals are described in detail in Table 12-2.

Table 12-2. DUART Signals—Detailed Signal Descriptions

Signal	I/O	Description	
UART_SIN[0:1]	I	Serial data in. Data is received on the receivers of UART0 and UART1 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, *rxcnt*, uses the frequency of the baud-rate generator to sample the data on SIN.

Table 12-2. DUART Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
UART_SOUT[0:1]	O	Serial data out. The serial data output signals for the UART0 and UART1 are set ('mark' condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on these signals, with the least significant bit transmitted first.	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation— An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to update and drive the data on SOUT.
$\overline{\text{UART\_CTS}}$[0:1]	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are 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 CCB clock.
$\overline{\text{UART\_RTS}}$[0:1]	O	Request to send. $\overline{\text{UART_RTS}}$x are active-low output signals that can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send ($\overline{\text{CTS}}$) 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 CCB clock.

12.3 Memory Map/Register Definition

Table 12-3 lists the DUART registers and their offsets. It lists the address, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 12-3.

There are two complete sets of DUART registers (one for UART0 and one for UART1). The two UARTs on the device are identical, except that the registers for UART0 are located at offset 0x4500 (local), and the registers for UART1 are located at offset 0x4600 (local). Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for either UART0 or UART1.

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 12.3.1.7, "Line Control Registers (ULCR0, ULCR1)," for more information on ULCR[DLAB].

All the DUART registers are one byte wide. Reads and writes to these registers must be byte-wide operations. Table 12-3 provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000–0xFFF are reserved.

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 12-3. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
Block Base Address: 0x0_4000				
0x500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	12.3.1.1/12-6
0x500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	12.3.1.2/12-6
0x500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	12.3.1.5/12-9
0x502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	12.3.1.6/12-11
0x502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	12.3.1.12/12-17
0x503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	12.3.1.7/12-12
0x504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	12.3.1.8/12-14
0x505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	12.3.1.9/12-14
0x506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	12.3.1.10/12-16
0x507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	12.3.1.11/12-17
0x510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	12.3.1.13/12-18
0x600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	12.3.1.1/12-6
0x600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	12.3.1.2/12-6
0x600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	12.3.1.5/12-9
0x602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	12.3.1.6/12-11
0x602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	12.3.1.12/12-17
0x603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	12.3.1.7/12-12
0x604	UMCR—ULCR[DLAB] = x UART1 modem control register	R/W	0x00	12.3.1.8/12-14
0x605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	12.3.1.9/12-14
0x606	UMSR—ULCR[DLAB] = x UART1 modem status register	R	0x00	12.3.1.10/12-16
0x607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	12.3.1.11/12-17
0x610	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	12.3.1.13/12-18

12.3.1 Register Descriptions

The following sections describe the UART0 and UART1 registers.

12.3.1.1 Receiver Buffer Registers (URBR0, URBR1) (ULCR[DLAB] = 0)

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, Section 12.3.1.9, "Line Status Registers (ULSR0, ULSR1)." Figure 12-3 shows the receiver buffer registers. Note that these registers have same offset as the UTHRs.

Figure 12-2 shows the bits in the URBRs.

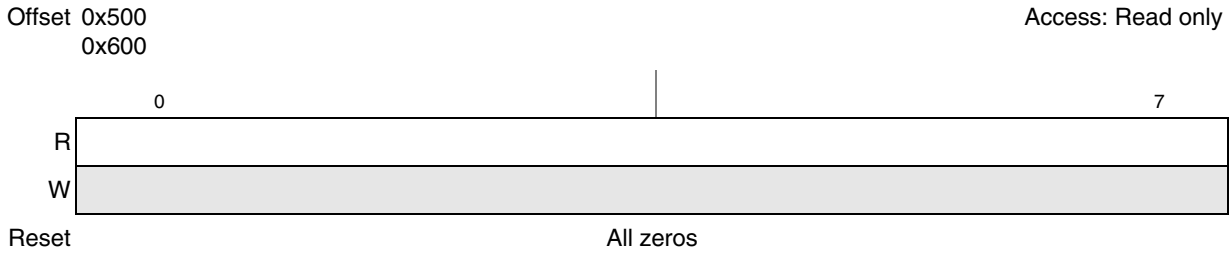


Figure 12-2. Receiver Buffer Registers (URBR0, URBR1)

Table 12-4 describes the fields of URBR.

Table 12-4. URBR Field Descriptions

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus (read only)

12.3.1.2 Transmitter Holding Registers (UTHR0, UTHR1) (ULCR[DLAB] = 0)

A write to these 8-bit registers causes the UART devices to transfer 5–8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR will be the first byte onto the bus. UDSR[$\overline{\text{TXRDY}}$] indicates when the FIFO is full. Refer to the Table 12-21 and the Table 12-22 for more details.

Figure 12-3 shows the bits in the UTHRs.

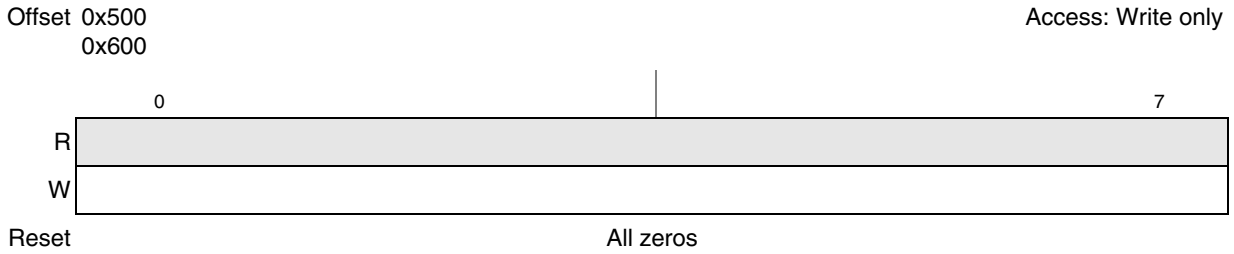


Figure 12-3. Transmitter Holding Registers (UTHR0, UTHR1)

Table 12-5 describes the fields of UTHR.

Table 12-5. UTHR Field Descriptions

Bits	Name	Description
0–7	DATA	Data that is written to UTHR (write only)

12.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB) (ULCR[DLAB] = 1)

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = 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 12-8.

Figure 12-4 shows the bits in the UDMBs.

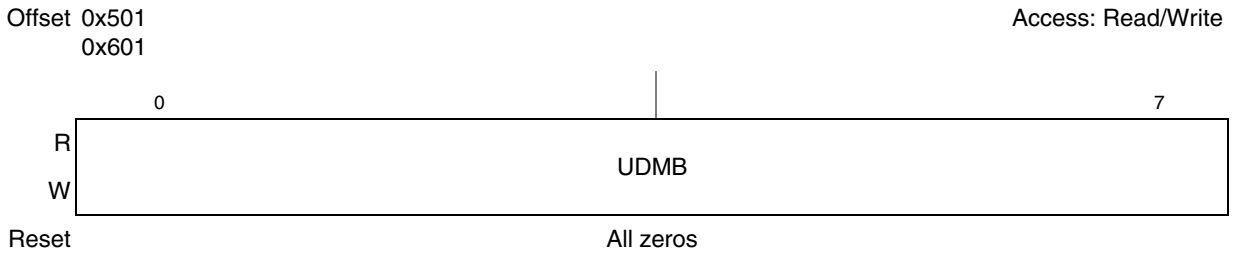


Figure 12-4. Divisor Most Significant Byte Registers (UDMB0, UDMB1)

Table 12-6 describes the fields of UDMB registers.

Table 12-6. UDMB Field Descriptions

Bits	Name	Description
0–7	UDMB	Divisor most-significant byte

Figure 12-5 shows the bits in the UDLBs.

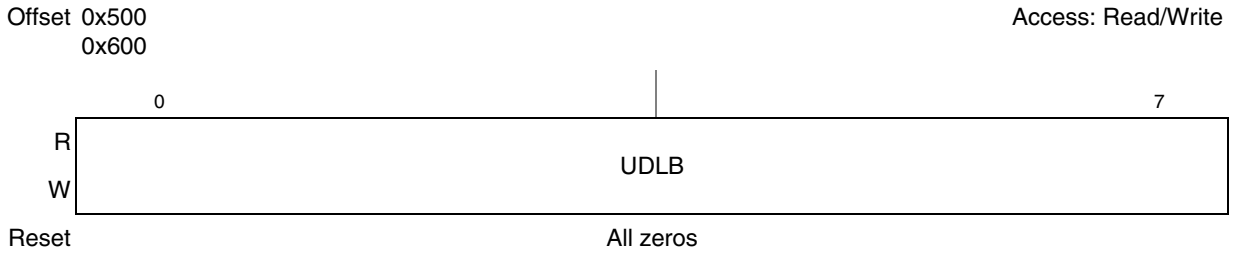


Figure 12-5. Divisor Least Significant Byte Registers (UDLB0, UDLB1)

Table 12-7 describes the fields of UDLB registers.

Table 12-7. UDLB Field Descriptions

Bits	Name	Description
0–7	UDLB	Divisor least-significant byte. This is concatenated with UDMB.

Table 12-8 shows baud rate when the input clock is at certain frequencies. Note that because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

Table 12-8. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		Input Clock (CCB) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	1736	6C8	266	9600.61444	0.0064
19,200	868	364	266	19,201.22888	0.0064
38,400	434	1B2	266	38,402.45776	0.0064
56,000	298	12A	266	55,928.41163	0.1280
128,000	130	82	266	128,205.12821	0.1600
256,000	65	41	266	256,410.25641	0.1600
9,600	2170	87A	333	9600.61444	0.0064
19,200	1085	43D	333	19,201.22888	0.0064
38,400	543	21F	333	38,367.09638	0.0858
56,000	372	174	333	56,003.58423	0.0064
128,000	163	A3	333	127,811.86094	0.1472
256,000	81	51	333	257,201.64609	0.4672

12.3.1.4 Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)

The UIER gives the user the ability to mask specific UART interrupts to the MPC8555E programmable interrupt controller (PIC).

Figure 12-6 shows the bits in the UIER.

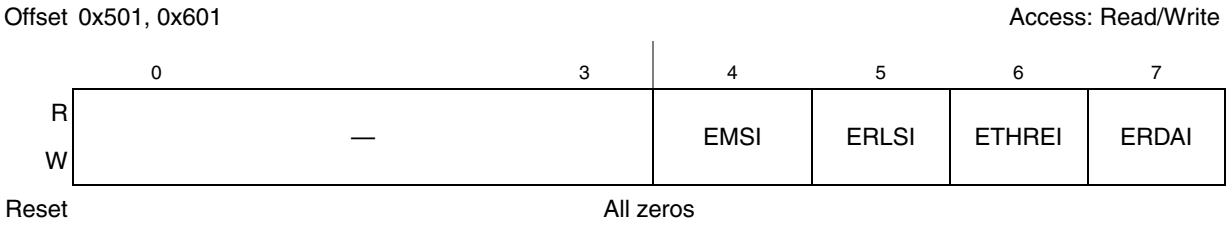


Figure 12-6. Interrupt Enable Register (UIER)

Table 12-9 describes the fields of UIER.

Table 12-9. 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 the clear-to-send bit in the UART modem status register (UMSR) 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 the FIFO mode

12.3.1.5 Interrupt ID Registers (UIIR0, UIIR1) (ULCR[DLAB] = 0)

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are:

1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty

4. Modem status

See Table 12-11 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 12-7 shows the bits in the UIIR.

Offset 0x502
0x602

Access: Read only

	0	1	2	3	4	5	6	7
R	FE		—		IID3	IID2	IID1	IID0
W								
Reset	0	0	0	0	0	0	0	1

Figure 12-7. Interrupt ID Registers (UIIR)

Table 12-10 describes the fields of the UIIR.

Table 12-10. 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 12-11. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5–6	IID2–1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 12-11.
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 12-11.

Table 12-11. UIIR IID Bits Summary

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	—	—	—	—
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.

Table 12-11. UIIR IID Bits Summary (continued)

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	Modem status	$\overline{\text{CTS}}$ input value changed since last read of UMSR	Read the UMSR.

12.3.1.6 FIFO Control Registers (UFCR0, UFCR1) (ULCR[DLAB] = 0)

The UFCR, a write-only register, is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

When the UFCR bits are written, the FIFO enable bit must also be set or else the UFCR bits are not programmed. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self-clearing bits.

Figure 12-8 shows the bits in the UFCRs.

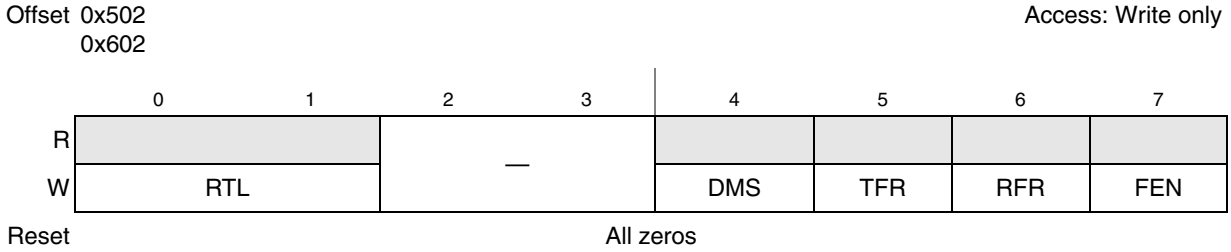


Figure 12-8. FIFO Control Registers (UFCR0, UFCR1)

Table 12-12 describes the fields of the UFCRs.

Table 12-12. 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 the designated interrupt trigger level as follows: 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3	—	Reserved

Table 12-12. UFCR Field Descriptions (continued)

Bits	Name	Description
4	DMS	DMA mode select. See Section 12.4.5.2, "DMA Mode Select," for more information. 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5	TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6	RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7	FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Enables the transmitter and receiver FIFOs

12.3.1.7 Line Control Registers (ULCR0, ULCR1)

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing the ULCR, the software should not re-write the ULCR when valid transfers on the UART bus are active. The software should not re-write the ULCR until the last STOP bit has been received and there are no new characters being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See Table 12-14 for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Figure 12-9 shows the bits in the ULCRs.

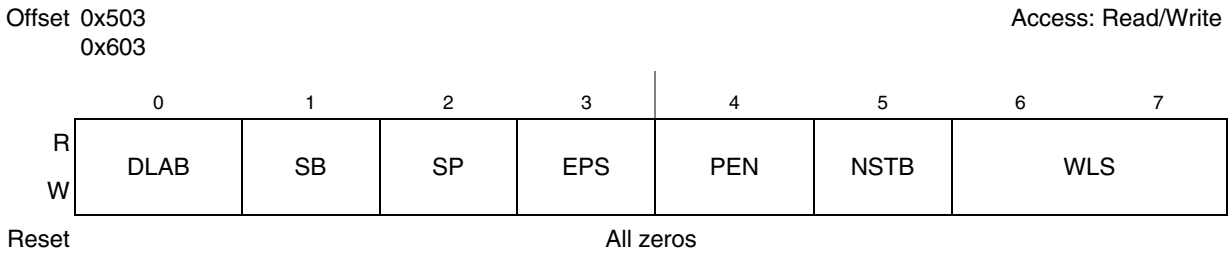


Figure 12-9. Line Control Register (ULCR)

Table 12-13 describes the fields of the ULCRs.

Table 12-13. 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 divisor latch least and most significant byte registers and alternate function register (UAFR)
1	SB	Set break. 0 Send normal UTHR data onto the serial output (SOUT) signal 1 Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected
2	SP	Stick parity. 0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
3	EPS	Even parity select. See Table 12-14 for more information. 0 If PEN = 1 and SP = 0, odd parity is selected. 1 If PEN = 1 and SP = 0, even parity is selected.
4	PEN	Parity enable. 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∫ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6–7	WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00 5 bits 01 6 bits 10 7 bits 11 8 bits

Table 12-14. 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

12.3.1.8 Modem Control Registers (UMCR0, UMCR1)

The UMCRs control the interface with the external peripheral device on the UART bus.

Figure 12-10 shows the bits in the UMCRs

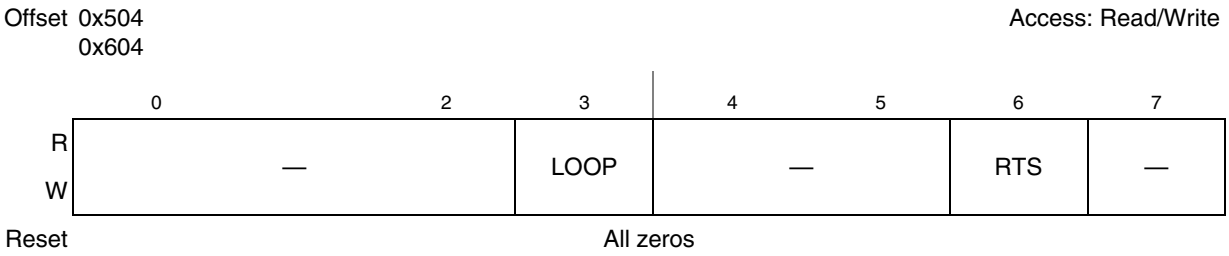


Figure 12-10. Modem Control Register (UMCR)

Table 12-15 describes the fields of UMCRs.

Table 12-15. UMCR Field Descriptions

Bits	Name	Description
0–2	—	Reserved.
3	LOOP	Local loopback mode. 0 Normal operation 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].
4–5	—	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.

12.3.1.9 Line Status Registers (ULSR0, ULSR1)

The ULSRs are read-only registers that monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

Figure 12-11 shows the bits in the ULSRs.

Offset 0x505
0x605

Access: Read only

	0	1	2	3	4	5	6	7
R	RFE	TEMT	THRE	BI	FE	PE	OE	DR
W								
Reset	0	1	1	0	0	0	0	0

Figure 12-11. Line Status Register (ULSR)

Table 12-16 describes the fields of the ULSRs.

Table 12-16. ULSR Field Descriptions

Bits	Name	Description
0	RFE	Receiver FIFO error. 0 This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1	TEMT	Transmitter empty. 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2	THRE	Transmitter holding register empty. 0 The UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3	BI	Break interrupt. 0 This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A break condition is expected to last at least two character lengths and 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. Note that the ULSR[BI] is set immediately after ULSR is read if bus remains zero and no mark state followed by a valid new character has been detected.
4	FE	Framing error. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then will receive the following new data.
5	PE	Parity error. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.

Table 12-16. ULSR Field Descriptions (continued)

Bits	Name	Description
6	OE	Overrun error. 0 This bit is cleared when ULSR is read. 1 Before the URBR is read, the URBR was overwritten with a new character. The old character is loss. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7	DR	Data ready. 0 This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character has been received in the URBR or the receiver FIFO.

12.3.1.10 Modem Status Registers (UMSR0, UMSR1)

The UMSRs track the status of the modem (or external peripheral device) clear to send ($\overline{\text{CTS}}$) signal for the corresponding UART.

Figure 12-12 shows the bits in the UMSRs.

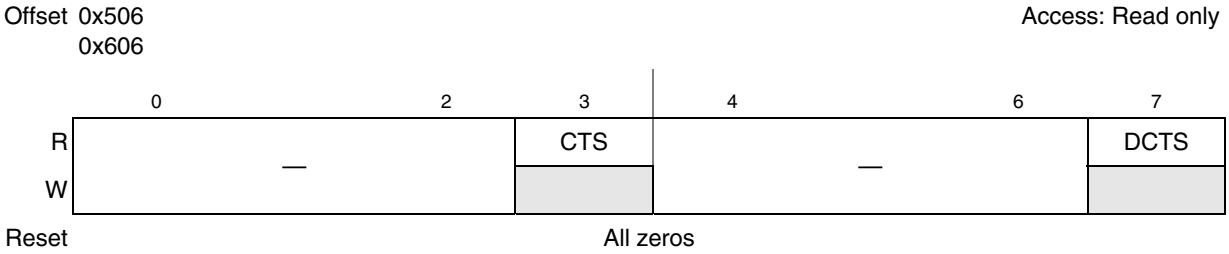


Figure 12-12. Modem Status Register (UMSR)

Table 12-17 describes the fields of the UMSRs.

Table 12-17. UMSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved.
3	CTS	Clear to send. Represents the inverted value of the $\overline{\text{CTS}}$ input pin from the external peripheral device 0 Corresponding $\overline{\text{CTS}}n$ is negated 1 Corresponding $\overline{\text{CTS}}n$ is asserted. The modem or peripheral device is ready for data transfers.
4–6	—	Reserved.
7	DCTS	Clear to send. 0 No change on the corresponding $\overline{\text{CTS}}n$ signal since the last read of UMSR[CTS] 1 The $\overline{\text{CTS}}n$ value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition

12.3.1.11 Scratch Registers (USCR0, USCR1)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

Figure 12-13 shows the bits in USCRs.

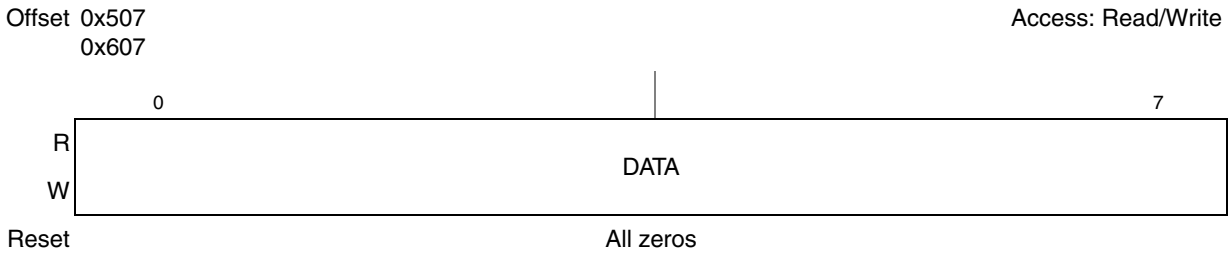


Figure 12-13. Scratch Register (USCR)

Table 12-18 describes the fields of the USCRs.

Table 12-18. USCR Field Descriptions

Bits	Name	Description
0–7	DATA	Data

12.3.1.12 Alternate Function Registers (UAFR0, UAFR1) (ULCR[DLAB] = 1)

The UAFRs give software the ability to write to both UART0 and UART1 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 12-14 shows the bits in the UAFRs.

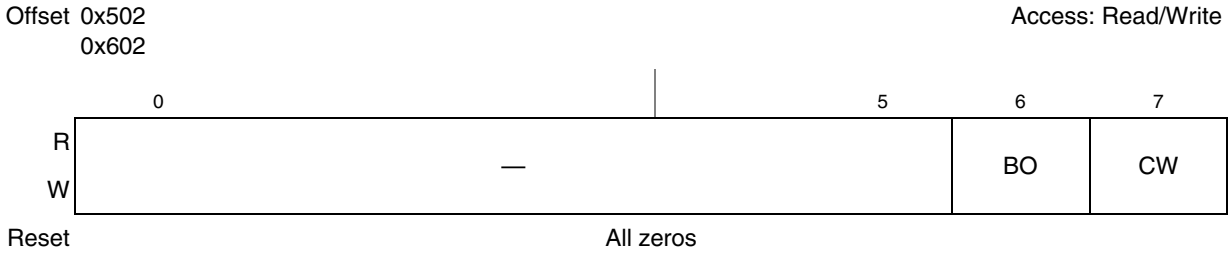


Figure 12-14. Alternate Function Register (UAFR)

Table 12-19 describes the fields of the UAFRs.

Table 12-19. UAFR Field Descriptions

Bits	Name	Description
0–5	—	Reserved.

Table 12-19. UAFR Field Descriptions (continued)

Bits	Name	Description
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 UART0 and UART1 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART0 is also a write to the corresponding register in UART1 and vice versa. The user needs to ensure that the LCR[DLAB] of both UARTs are in the same state before executing a concurrent write to register addresses 0x500, 0x501 and 0x502.

12.3.1.13 DMA Status Registers (UDSR0, UDSR1)

The DMA status registers (UDSRs) are read-only registers that return transmitter and receiver FIFO status. UDSRs also provide the ability to assist DMA data operations to and from the FIFOs.

Figure 12-15 shows the bits in UDSRs.

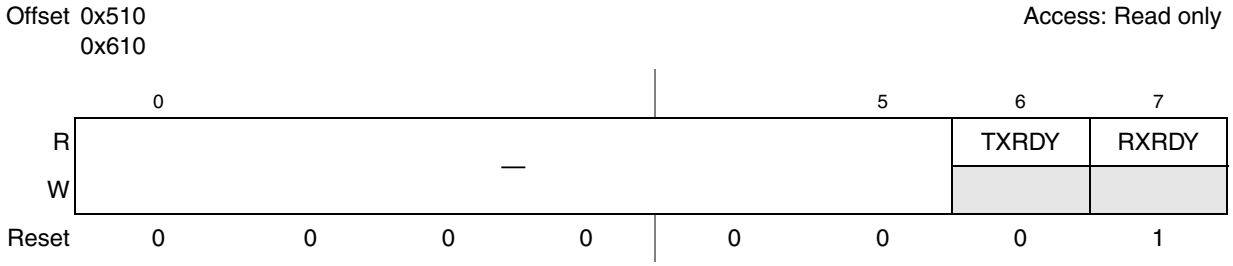


Figure 12-15. DMA Status Register (UDSR)

Table 12-20 describes the fields of the UDSRs.

Table 12-20. UDSR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 12-22. 1 This bit is set, as shown in Table 12-21.
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 the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 12-24. 1 This bit is set, as shown in Table 12-23.

Table 12-21. 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 12-22. 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 when the transmitter FIFO is not yet full.

Table 12-23. 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 12-24. 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.

12.4 Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock signal.

The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see Section 12.4.5, “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.

12.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 12-16. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.

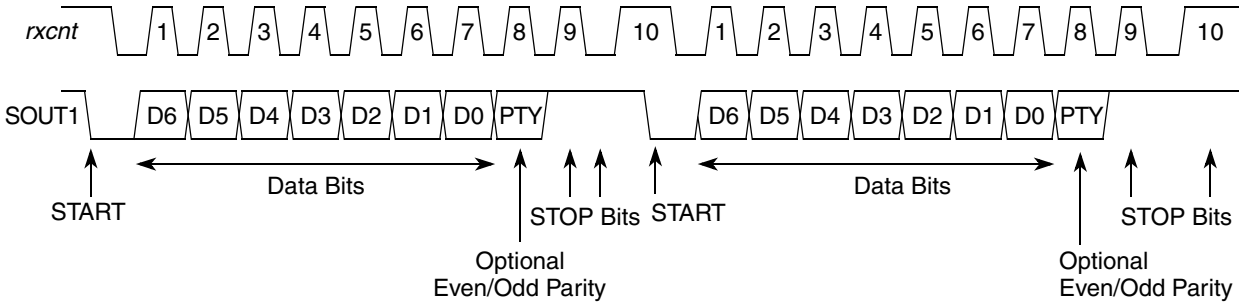


Figure 12-16. UART Bus Interface Transaction Protocol Example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer bits (least-significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loopback mode, different errors, and FIFO mode.

12.4.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 12-16 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 the UART line control register (ULCR).When the bus is idle, SOUT is high.

12.4.1.2 Data Transfer

Each data transfer contains 5–8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time a START bit is generated followed by 5–8 of the data bits previously written to the UTHR. The data bits are driven from the least significant to the most significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to the UTHR.

12.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see Section 12.3.1.7, "Line Control Registers (ULCR0, ULCR1)." Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data a parity error can occur if an unexpected parity value is detected. (See Section 12.3.1.9, "Line Status Registers (ULSR0, ULSR1).")

12.4.1.4 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.

12.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the CCB clock input and dividing the input by any divisor from 1 to $2^{16} - 1$.

The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

$$\text{Baud rate} = (1/16) \times (\text{CCB clock frequency/divisor value})$$

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

12.4.3 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).

12.4.4 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.

12.4.4.1 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.

12.4.4.2 Parity Error

A parity error occurs, and ULSR[PE] is set, when unexpected parity values are encountered while receiving data. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

12.4.4.3 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.

12.4.5 FIFO Mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (UFCR) is used to enable and clear the receiver and transmitter FIFOs

and set the FIFO receiver trigger level UFCR[RTL] to control the received data available interrupt UIER[ERDAI].

The UFCR also selects the type of DMA signaling. The UDSR[RXRDY] indicates the status of the receiver FIFO. The DMA status registers (UDSR[TXRDY]) indicate when the transmitter FIFO is full. When in FIFO mode, data written to UTHR is placed into the transmitter FIFO. The first byte written to UTHR is the first byte onto the UART bus.

12.4.5.1 FIFO Interrupts

In FIFO mode, the UIER[ERDAI] is set when a time-out interrupt occurs. When a receive data time-out occurs there is a maskable interrupt condition (through UIER[ERDAI]). See Section 12.3.1.4, "Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)," for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. Interrupt ID3 UIIR[IID3] bit is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by UIIR[IID]) is cleared when the URBR is read. See Section 12.3.1.5, "Interrupt ID Registers (UIIR0, UIIR1) (ULCR[DLAB] = 0)," for more information.

The UIIR[FE] bits indicate if FIFO mode is enabled.

12.4.5.2 DMA Mode Select

The UDSR[RXRDY] bit reflects the status of the receiver FIFO or URBR. In mode 0 (UFCR[DMS] is cleared), UDSR[RXRDY] is cleared when there is at least one character in the receiver FIFO or URBR and it is set when there are no more characters in the receiver FIFO or URBR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[RXRDY] is cleared when the trigger level or a time-out has been reached and it is set when there are no more characters in the receiver FIFO.

The UDSR[TXRDY] bit reflects the status of the transmitter FIFO or UTHR. In mode 0 (UFCR[DMS] is cleared), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set after the first character is loaded into the transmitter FIFO or UTHR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set when the transmitter FIFO is full.

See Section 12.3.1.13, "DMA Status Registers (UDSR0, UDSR1)," for a complete description of the USDR[RXRDY] and USDR[TXRDY] bits.

12.4.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 0 (UIIR[0]), is cleared. The interrupt enable register (UIER) is used to mask specific interrupt types. For more details refer to the description of UIER in Section 12.3.1.4, "Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)."

When the interrupts are disabled in UIER, polling software can not use UIIR[0] to determine whether the UART is ready for service. The software must monitor the appropriate bits in the line status (ULSR) and/or the modem status (UMSR) registers. UIIR[0] can be used for polling if the interrupts are enabled in UIER.

12.5 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-wide operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. 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 13
Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), SDRAM machine, and user-programmable machines (UPMs) of the LBC. Finally, it includes an initialization and applications information section with many specific examples of its use.

13.1 Introduction

Figure 13-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and SDRAM controller.

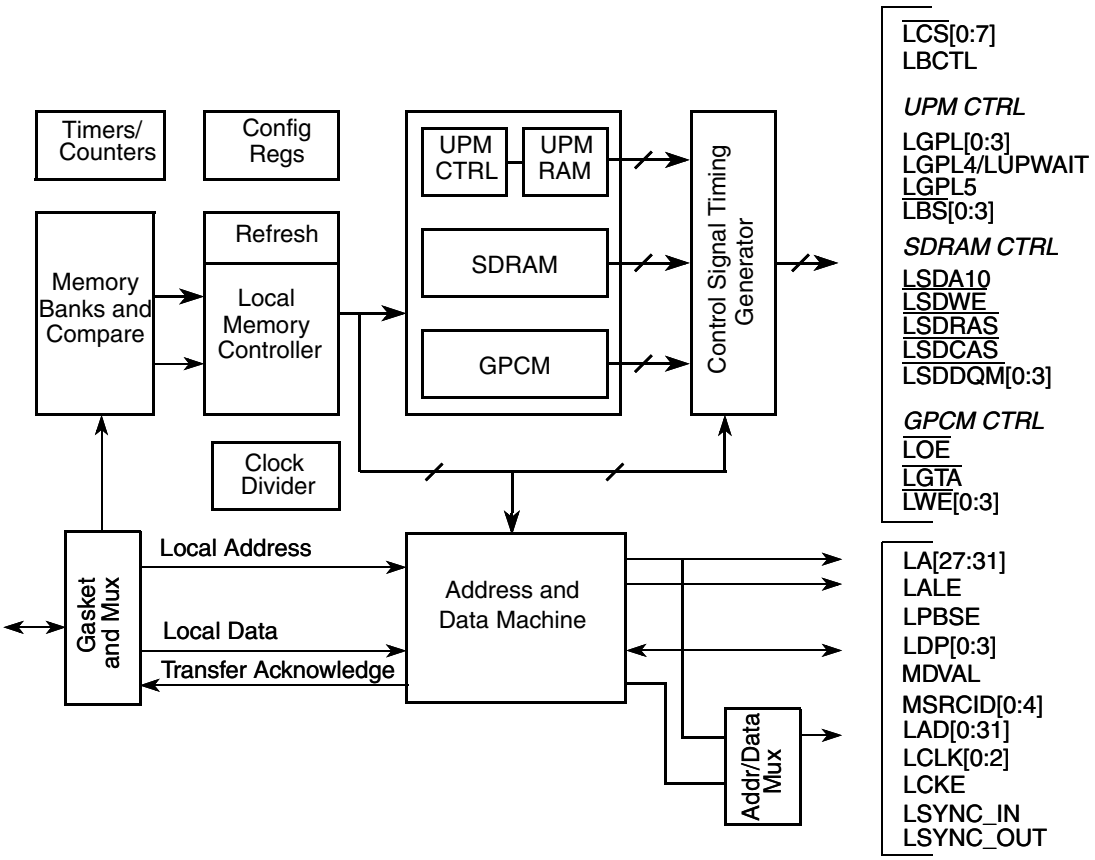


Figure 13-1. Local Bus Controller Block Diagram

13.1.1 Overview

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 synchronous DRAM (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.

13.1.2 Features

The LBC main features are as follows:

- Memory controller with eight memory banks
  - 32-bit[1] address decoding with mask
  - Variable memory block sizes (32 Kbytes to 4 Gbytes)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - Up to 256-byte bursts, arbitrarily aligned
  - 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 four concurrent open pages per device
  - Supports SDRAM port size of 32, 16, and 8 bits
  - Supports external address and/or command lines 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 3-clock access to external devices

1.Refers to the logical address space of the LBC. Once the address is decoded by the LBC, it is the rightmost 32-bits which are used for the transaction (32-bit physical address space).

  - 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

13.1.3 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.

13.1.3.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 2, 4, and 8 between the faster system (CCB) clock and the slower external bus clock (LCLK[0:2]). This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]). In addition to establishing the frequency of the external local bus clock, CLKDIV also 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.

13.1.3.2 Source ID Debug Mode

In debug mode, the LBC provides the ID of a transaction source on external device signals. This mode is enabled on power-on reset, as described in Section 20.4.4, "Local Bus Interface Debug." When placed in this mode, the 5-bit internal ID of the current transaction source appears on MSRCID[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 (MDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on MSRCID[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, lsbs} where row corresponds to the same row address for the given column address and lsbs are the unconnected lsbs of the address for a given port size.
- If a valid source ID is detected on MSRCID[0:4] and MDVAL is asserted, valid data may be latched from LAD[0:31].

13.1.4 Power-Down Mode

The LBC can enter a power-down mode when the system stops the internal (system) clock to the block by using a handshake protocol initiated by the DEVDISR[LBC] setting in the global utilities block. On entering power-down mode, the LBC places any SDRAM devices, if used, in self-refresh mode before the bus clock is stopped. The LBC also allows the DLL sufficient time to recover following the reapplication of the system clock.

Once the LBC has been put into power-down mode, the only way to exit from this mode is through $\overline{\text{HRESET}}$.

13.2 External Signal Descriptions

Table 13-1 contains a list of external signals related to the LBC and summarizes their function. I/O impedance of designated local bus signals is determined by PORIMPSCR, as described in Section 18.4.1.3, "POR I/O Impedance Status and Control Register (PORIMPSCR)."

Table 13-1. Signal Properties—Summary

Name	Number of Signals	Direction	Function
LALE	1	Output	External address latch enable
$\overline{\text{LCS}}$	8	Output	Chip selects
$\overline{\text{LWE}}n$/ $\overline{\text{LSDDQM}}n$/ $\overline{\text{LBS}}n$	4	Output Output Output	GPCM mode: write enable SDRAM mode: byte lane data mask UPM mode: byte (lane) select
LSDA10/ LGPL0	1	Output Output	SDRAM mode: row address bit/command bit UPM mode: general-purpose line 0
$\overline{\text{LSDWE}}$/ LGPL1	1	Output Output	SDRAM mode: write enable UPM mode: general-purpose line 1

Table 13-1. Signal Properties—Summary (continued)

Name	Number of Signals	Direction	Function
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	1	Output Output Output	GPCM mode: output enable SDRAM mode: row address strobe UPM mode: general-purpose line 2
$\overline{\text{LSDCAS}}$/ LGPL3	1	Output Output	SDRAM mode: column address strobe UPM mode: general-purpose line 3
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT/ LPBSE	1	Input Output Input Output	GPCM mode: transaction termination UPM mode: general-purpose line 4 UPM mode: external device wait Local bus parity byte select
LGPL5	1	Output	UPM mode: general-purpose line 5
LBCTL	1	Output	Data buffer control
LA[27:31]	5	Output	Local bus non-multiplexed address lsbs
LAD[0:31]	32	Input/Output	Multiplexed address/data bus
LDP	4	Input/Output	Local bus data parity
LCKE	1	Output	Local bus clock enable
LCLK[0:2]	3	Output	Local bus clocks
LSYNC_IN	1	Input	DLL synchronize input
LSYNC_OUT	1	Output	DLL synchronize output
MDVAL	1	Output	In LBC debug mode: local bus data valid
MSRCID	5	Output	In LBC debug mode: local bus source ID

Table 13-2 contains detailed external signal descriptions for the LBC.

Table 13-2. 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.	
		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 LBCR[AHD] field. Note that no other control signals are asserted during the assertion of LALE.
$\overline{\text{LCS}}$[0:7]	O	Chip selects. Eight chip selects are provided which are mutually exclusive.	
		State Meaning	Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. $\overline{\text{LCS}}$[0:7] are provided on a per-bank basis with $\overline{\text{LCS0}}$ corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LWE}}$[0:3]/ $\overline{\text{LSDDQM}}$[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 BR*n*[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, $\overline{\text{LSDDQM}}$[0:3] function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. $\overline{\text{LSDDQM}}$[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 13.4.4.4, "RAM Array," for programming details about $\overline{\text{LBS}}$[0:3].
		Timing	Assertion/Negation—See Section 13.4.2, "General-Purpose Chip-Select Machine (GPCM)," for details regarding the timing of $\overline{\text{LWE}}$[0:3].
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.
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	O	GPCM output enable/SDRAM $\overline{\text{RAS}}$/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 ($\overline{\text{RAS}}$). 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 $\overline{\text{CAS}}$/General-purpose line 3	
		State Meaning	Asserted/Negated—In SDRAM mode, drives the column address strobe ($\overline{\text{CAS}}$). One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT/ LPBSE	I/O	GPCM transfer acknowledge/General-purpose line 4/UPM wait/parity byte select	
		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. 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 RMW-parity devices.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LGPL5	O	General-purpose line 5	
		State Meaning	Asserted/Negated—One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
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 OR*n*[BCTLD].	
		State Meaning	Asserted/Negated—The LBCTL signal 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.
LA[27:31]	O	Local bus non-multiplexed address lsbs. 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.	
		State Meaning	Asserted/Negated—Although the LBC shares an address and data bus, up to five lsbs 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 lsbs 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.
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 LDP3 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—LCKE is the bus clock enable signal (CKE) for JEDEC-standard SDRAM devices. Asserted during normal SDRAM operation.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LCLK[0:2]	O	Local bus clocks	
		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 13-19 on page 13-31), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:31] and $\overline{\text{LCS}n}$) by a time delay matching the delay of the DLLtiming loop set up between LSYNC_OUT and LSYNC_IN.
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[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.
MDVAL	O	Local bus data valid (LBC debug mode only)	
		State Meaning	Asserted/Negated—For a read, MDVAL asserts for one bus cycle in the cycle immediately preceding the sampling of read data on LAD[0:31]. For a write, MDVAL asserts for one bus cycle during the final cycle for which the current write data on LAD[0:31] is valid. During burst transfers, MDVAL asserts for each data beat.
		Timing	Assertion/Negation—Valid only while the LBC is in system debug mode. In debug mode, MDVAL asserts when the LBC generates a data transfer acknowledge.
MSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all MSRCID[0:4] signals are driven high unless MSRCID[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 MDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, MSRCID[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, MSRCID[0:4] is valid only during $\overline{\text{CAS}}$ phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits are all present on LAD[0:31] during a $\overline{\text{CAS}}$ cycle

13.3 Memory Map/Register Definition

Table 13-3 shows the memory mapped registers of the LBC and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset listed in Table 13-3. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

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 13-3. Local Bus Controller Memory Map

Offset	Use	Access	Reset	Section/Page
Local Bus Block Base Address: 0x0_6000				
0x000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	13.3.1.1/13-10
0x008	BR1—Base register 1		0x0000_0000	
0x010	BR2—Base register 2			
0x018	BR3—Base register 3			
0x020	BR4—Base register 4			
0x028	BR5—Base register 5			
0x030	BR6—Base register 6			
0x038	BR7—Base register 7			
0x004	OR0—Options register 0	R/W	0x0000_0FF7	13.3.1.2/13-12
0x00C	OR1—Options register 1		0x0000_0000	
0x014	OR2—Options register 2			
0x01C	OR3—Options register 3			
0x024	OR4—Options register 4			
0x02C	OR5—Options register 5			
0x034	OR6—Options register 6			
0x03C	OR7—Options register 7			
0x068	MAR—UPM address register	R/W	0x0000_0000	13.3.1.3/13-17
0x070	MAMR—UPMA mode register	R/W	0x0000_0000	13.3.1.4/13-18
0x074	MBMR—UPMB mode register	R/W	0x0000_0000	13.3.1.4/13-18
0x078	MCMR—UPMC mode register	R/W	0x0000_0000	13.3.1.4/13-18
0x084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	13.3.1.5/13-20
0x088	MDR—UPM data register	R/W	0x0000_0000	13.3.1.6/13-21
0x094	LSDMR—SDRAM mode register	R/W	0x0000_0000	13.3.1.7/13-21
0x0A0	LURT—UPM refresh timer	R/W	0x0000_0000	13.3.1.8/13-23
0x0A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	13.3.1.9/13-24
0x0B0	LTESR—Transfer error status register	w1c	0x0000_0000	13.3.1.10/13-25
0x0B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	13.3.1.11/13-26

Table 13-3. Local Bus Controller Memory Map (continued)

Offset	Use	Access	Reset	Section/Page
0x0B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	13.3.1.12/13-27
0x0BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	13.3.1.13/13-28
0x0C0	LTEAR—Transfer error address register	R/W	0x0000_0000	13.3.1.14/13-29
0x0D0	LBCR—Configuration register	R/W	0x0000_0000	13.3.1.15/13-29
0x0D4	LCRR—Clock ratio register	R/W	0x8000_0008	13.3.1.16/13-31

1 Port size for BR0 is configured from external signals during reset, hence '*nn*' is either 0x08, 0x10, or 0x18.

13.3.1 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 13-3 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.

Bits designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

13.3.1.1 Base Registers (BR0–BR7)

The base registers (BR*n*), shown in Figure 13-2, contain the base address and address types for each memory bank. The memory controller uses this information to compare the address bus value with the current address accessed. Each register (bank) includes a memory attribute and selects the machine for memory operation handling. Note that after system reset, BR0[V] is set, BR1[V]–BR7[V] are cleared, and the value of BR0[PS] reflects the initial port size configured by the boot ROM location signals.

Offset 0x000 (BR0), 0x008 (BR1), 0x010 (BR2), 0x018 (BR3), 0x020 (BR4), 0x028 (BR5), 0x030 (BR6), 0x038 (BR7) — Access: Read/Write

	0								16	17	18	19	20	21	22	23	24		26	27	28	29	30	31
R W	BA								—			PS	DECC			WP	MSEL			—	ATOM		—	V
Reset BR0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1[1]
Reset BR1–BR7	All zeros																							

1 BR0 has its valid bit set during reset. Thus bank 0 is valid with the port size (PS) configured from external boot ROM configuration signals during reset. All other base registers have all bits cleared to zero during reset.

Figure 13-2. Base Registers (BR*n*)

Table 13-4 describes BR*n* fields.

Table 13-4. BR*n* Field Descriptions

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. Used with the address mask bits OR*n*[AM].
17–18	—	Reserved
19–20	PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the boot ROM location signals during reset. For all other banks the value is reset to 00 (port size not defined). 00 Reserved 01 8-bit 10 16-bit 11 32-bit
21–22	DECC	Specifies the method for data error checking. 00 Data error checking disabled, but normal parity generation 01 Normal parity generation and checking 10 Read-modify-write parity generation and normal parity checking (32-bit port size only) 11 Reserved
23	WP	Write protect 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert $\overline{\text{LCS}n}$ 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. $\overline{\text{LCS}n}$ 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.

13.3.1.2 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
- UPM mode
- SDRAM mode

The OR*n* registers are interpreted differently depending on which of the three machine types is selected for that bank.

13.3.1.2.1 Address Mask

The address mask field of the option registers (OR*n*[AM]) mask up to 17 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 13-5 shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-5. Memory Bank Sizes in Relation to Address Mask

Address Mask	Memory Bank Size	Address Mask	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes	1111_1111_1000_0000_0	8 Mbytes
1000_0000_0000_0000_0	2 Gbytes	1111_1111_1100_0000_0	4 Mbytes
1100_0000_0000_0000_0	1 Gbyte	1111_1111_1110_0000_0	2 Mbytes
1110_0000_0000_0000_0	512 Mbytes	1111_1111_1111_0000_0	1 Mbyte
1111_0000_0000_0000_0	256 Mbytes	1111_1111_1111_1000_0	512 Kbytes
1111_1000_0000_0000_0	128 Mbytes	1111_1111_1111_1100_0	256 Kbytes
1111_1100_0000_0000_0	64 Mbytes	1111_1111_1111_1110_0	128 Kbytes
1111_1110_0000_0000_0	32 Mbytes	1111_1111_1111_1111_0	64 Kbytes
1111_1111_0000_0000_0	16 Mbytes	1111_1111_1111_1111_1	32 Kbytes

13.3.1.2.2 Option Registers (OR*n*)—GPCM Mode

Figure 13-3 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the GPCM machine.

Offset 0x004 (OR0), 0x00C (OR1), 0x014 (OR2), 0x01C (OR3), 0x024 (OR4), 0x02C (OR5), 0x034 (OR6), 0x03C (OR7) — Access: Read/Write

	0–16	17–18	19	20	21	22	23	24–27	28	29	30	31
R/W	AM	—	BCTLD	CSNT	ACS		XACS	SCY	SETA	TRLX	EHTR	EAD
Reset OR0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0	0	1	1	1	1	1 1 1 1	0	1	1	1[1]
Reset OR1–7	All zeros											

Figure 13-3. Option Registers (OR*n*) in GPCM Mode

[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.

Table 13-6 describes OR*n* fields for GPCM mode.

Table 13-6. OR*n*—GPCM Field Descriptions

Bits	Name	Description
0–16	AM	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. 0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses.
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	CSNT	Chip select negation time. Determines when $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only $\overline{\text{LWE}}$ is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally. 1 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated subject to the value of LCRR[CLKDIV].

LCRR [CLKDIV]	CSNT	Meaning
x	0	$\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally.
2	1	$\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally.
4 or 8	1	$\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated quarter of a bus clock cycle earlier.

Table 13-6. OR*n*—GPCM Field Descriptions (continued)

Bits	Name	Description
21–22	ACS	Address to chip-select setup. Determines the delay of the $\overline{\text{LCS}n}$ assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11.

LCRR [CLKDIV]	Value	Meaning
x	00	$\overline{\text{LCS}n}$ 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	$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.
	11	$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.
4 or 8	10	$\overline{\text{LCS}n}$ is output a quarter bus clock cycle after the address lines.
	11	$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.

23	XACS	Extra address to chip-select setup. Setting this bit increases the delay of the $\overline{\text{LCS}n}$ assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 1. 0 Address to chip-select setup is determined by ORx[ACS] and LCRR[CLKDIV]. 1 Address to chip-select setup is extended (see Table 13-23 and Table 13-24 for LCRR[CLKDIV] = 4 or 8, Table 13-25 and Table 13-26 for LCRR[CLKDIV] = 2).
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: • Adds an additional cycle between the address and control signals (only if ACS ≠ 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 • $\overline{\text{LCS}n}$ (only if ACS ≠ 00) and $\overline{\text{LWE}}$ signals are negated one cycle earlier during writes.

Table 13-6. OR*n*—GPCM 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. 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]).

13.3.1.2.3 Option Registers (OR*n*)—UPM Mode

Figure 13-4 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects a UPM machine.

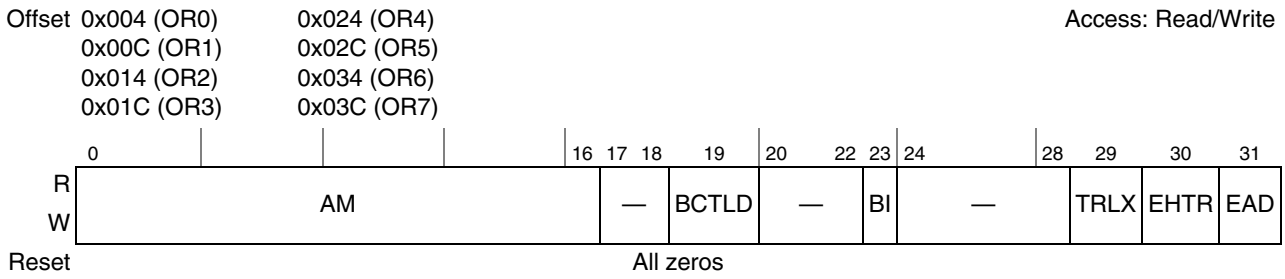


Figure 13-4. Option Registers (OR*n*) in UPM Mode

Table 13-7 describes BR*n* fields for UPM mode.

Table 13-7. 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

Table 13-7. OR*n*—UPM Field Descriptions (continued)

Bits	Name	Description
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.
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. **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]).

13.3.1.2.4 Option Registers (OR*n*)—SDRAM Mode

Figure 13-5 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the SDRAM machine.

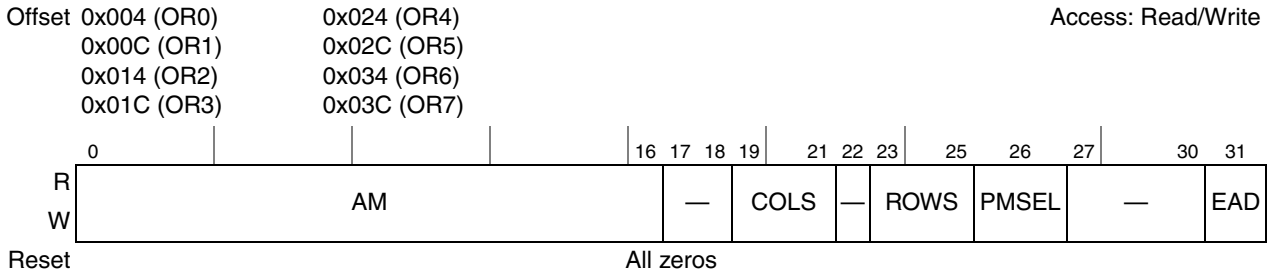


Figure 13-5. Option Registers (OR*n*) in SDRAM Mode

Table 13-8 describes BR*n* fields for SDRAM mode.

Table 13-8. 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

Table 13-8. OR*n*—SDRAM Field Descriptions (continued)

Bits	Name	Description
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
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]).

13.3.1.3 UPM Memory Address Register (MAR)

Figure 13-6 shows the fields of the UPM memory address register (MAR).

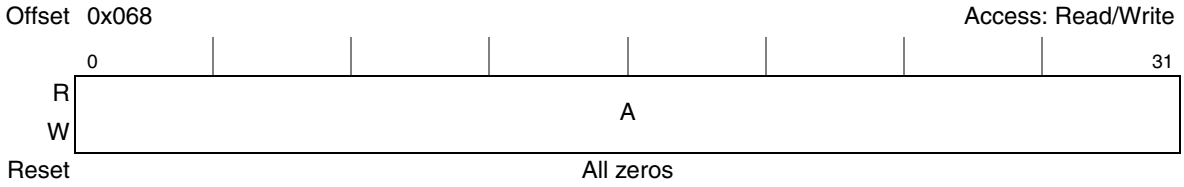


Figure 13-6. UPM Memory Address Register (MAR)

Table 13-9 describes the MAR fields.

Table 13-9. 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.

13.3.1.4 UPM Mode Registers (M*x*MR)

The UPM machine mode registers (MAMR, MBMR and MCMR), shown in Figure 13-7, contain the configuration for the three UPMs.

Offset 0x070 (MAMR)
0x074 (MBMR)
0x078 (MCMR)

Access: Read/Write

	0	1	2 3	4	5 7	8 9	10	12 13	14 17	18 21	22 25	26 31
R W	—	RFEN	OP	UWPL	AM	DS	G0CL	GPL4	RLF	WLF	TLF	MAD
Reset	All zeros											

Figure 13-7. UPM Mode Registers (M*x*MR)

Table 13-10 describes UPM mode fields.

Table 13-10. M*x*MR 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. See Section 13.4.4.2, “Programming the UPMs,” for important programming considerations. 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 13-10. M*x*MR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>5–7</td><td>AM</td><td>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>
<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>
<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</td><td colspan="8">Reserved</td></tr>
<tr><td>111</td><td colspan="8">Reserved</td></tr>
</table></td></tr>
<tr><td>8–9</td><td>DS</td><td>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</td></tr>
<tr><td>10–12</td><td>G0CL</td><td>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</td></tr>
<tr><td>13</td><td>GPL4</td><td>LGPL4 output line disable. Determines how the LGPL4/LUPWAIT signal is controlled by the corresponding bits in the UPMn array. See Table 13-30.
<table>
<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>
<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>
</table></td></tr>
</table>

Table 13-10. M*x*MR Field Descriptions (continued)

Bits	Name	Description
14–17	RLF	Read loop field. Determines the number of times a loop defined in the UPM*n* will be executed for a burst- or single-beat read pattern or when M*x*MR[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 UPM*n* will be 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 UPM*n* will be 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 UPM*n*.

13.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register (MRTPR), shown in Figure 13-8, is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

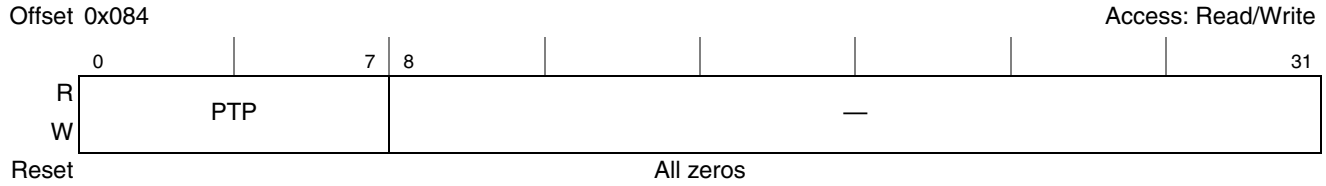


Figure 13-8. Memory Refresh Timer Prescaler Register (MRTPR)

Table 13-11 describes MRTPR fields.

Table 13-11. 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 0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

13.3.1.6 UPM Data Register (MDR)

The memory data register (MDR), shown in Figure 13-9, 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.

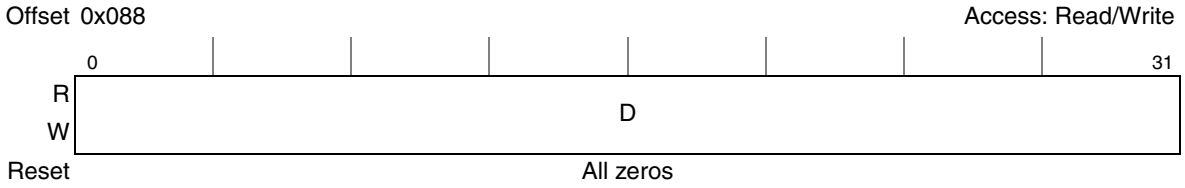


Figure 13-9. UPM Data Register (MDR)

Table 13-12 describes MDR[D].

Table 13-12. MDR Field Descriptions

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 (M*x*MR[OP] = 01 or M*x*MR[OP] = 10).

13.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR), shown in Figure 13-10, is used to configure operations pertaining to SDRAM.

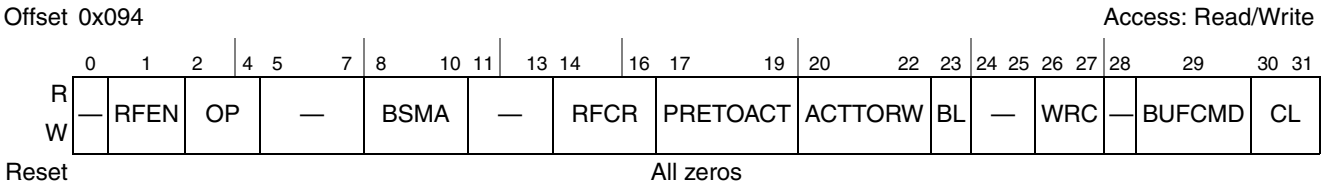


Figure 13-10. SDRAM Machine Mode Register (LSDMR)

Table 13-12 describes LSDMR fields.

Table 13-13. LSDMR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>—</td><td>Reserved</td></tr>
<tr><td>1</td><td>RFEN</td><td>Refresh enable. Indicates that the SDRAM requires refresh services.
0 Refresh services are not required.
1 Refresh services are required.</td></tr>
<tr><td>2–4</td><td>OP</td><td>SDRAM operation. Selects the operation that occurs when the SDRAM device is accessed.
<table>
<tr><th>Value</th><th>Meaning</th><th>Use</th></tr>
<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>Power-down mode or 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>
</table></td></tr>
<tr><td>5–7</td><td>—</td><td>Reserved</td></tr>
<tr><td>8–10</td><td>BSMA</td><td>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.
000 LA12:LA13 100 LA16:LA17
001 LA13:LA14 101 LA17:LA18
010 LA14:LA15 110 LA18:LA19
011 LA15:LA16 111 LA19:LA20</td></tr>
<tr><td>11–13</td><td>—</td><td>Reserved</td></tr>
<tr><td>14–16</td><td>RFCR</td><td>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.
000 Reserved 100 6 clocks
001 3 clocks 101 7 clocks
010 4 clocks 110 8 clocks
011 5 clocks 111 16 clocks</td></tr>
<tr><td>17–19</td><td>PRETOACT</td><td>Defines the earliest timing for ACTIVATE or REFRESH command after a PRECHARGE command (number of bus clock cycle wait states).
000 8 100 4
001 1 101 5
010 2 110 6
011 3 111 7</td></tr>
</table>

Table 13-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
20–22	ACTTORW	Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). 000 8 100 4 001 Reserved 101 5 010 2 110 6 011 3 111 7
23	BL	Sets the burst length for SDRAM accesses. 0 SDRAM burst length is 4. Use this value if the device port size is16. 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.
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. 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 $\overline{\text{LCS}n}$, LCKE, LALE, and $\overline{\text{LSDDQM}}$[0:3] to be asserted for LCRR[BUFCMDC] cycles, instead of one. 0 Normal timing for the control lines 1 All control lines except $\overline{\text{LCS}n}$ 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 13-22. 01 1 10 2 11 3

13.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT), shown in Figure 13-11, generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled (M*x*MR[RFEN] = 1). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

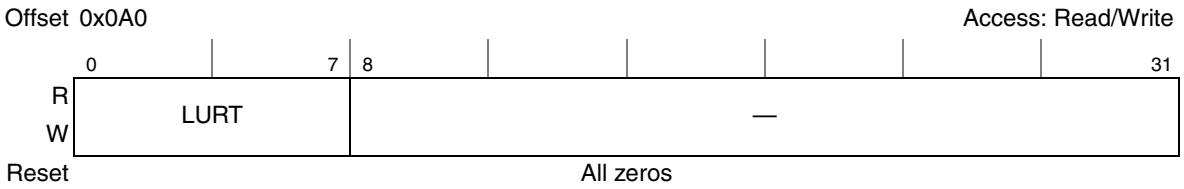


Figure 13-11. UPM Refresh Timer (LURT)

Table 13-14 describes LURT fields.

Table 13-14. LURT Field Descriptions

Bits	Name	Description
0–7	LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LURT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ **Example**: For a 266-MHz system clock and a required service rate of 15.6 µs, given MRTPR[PTP] = 32, the LURT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value (0x00) sets the maximum period to 256 × MRTPR[PTP] system clock cycles.
8–31	—	Reserved

13.3.1.9 SDRAM Refresh Timer (LSRT)

The SDRAM refresh timer (LSRT), shown in Figure 13-12, generates a refresh request for all valid banks that selected a SDRAM machine and are refresh-enabled (LSDMR[RFEN] = 1). Each time the timer expires, all qualifying banks generate a bank staggering auto-refresh request using the SDRAM machine.

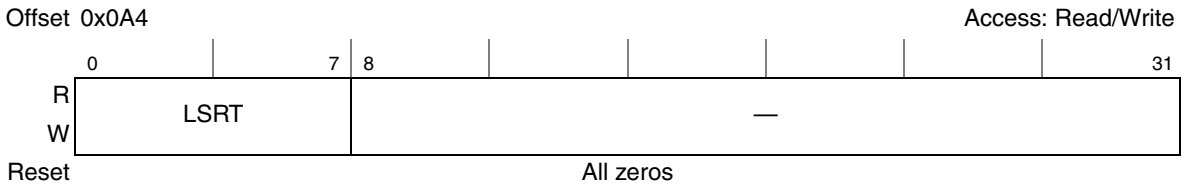


Figure 13-12. LSRT SDRAM Refresh Timer (LSRT)

Table 13-15 describes LSRT fields.

Table 13-15. LSRT Field Descriptions

Bits	Name	Description
0–7	LSRT	SDRAM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LSRT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ **Example:** For a 266-MHz system clock and a required service rate of 15.6 µs, given PTP = 32, the LSRT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value, 0x00, sets the maximum period to 256 × MRTPR[PTP] system clock cycles.
8–31	—	Reserved

13.3.1.10 Transfer Error Status Register (LTESR)

The LBC has the following registers for error management:

- The transfer error status register (LTESR) indicates the cause of an error.
- The transfer error check disable register (LTEDR) is used to enable (and disable) error checking.
- The transfer error check interrupt register (LTEIR) enables reporting of errors through an interrupt.
- The transfer error attributes register (LTEATR) captures source attributes of an error.
- The transfer error address register (LTEAR) captures the address of a transaction that caused an error.

LTESR, shown in Figure 13-13, is a write-one-to-clear register. Reading LTESR occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written and the data in the corresponding bit location is a 1. For example, to clear only the write protect error bit (LTESR[WP]) without affecting other LTESR bits, 0x0400_0000 should be written to the register. Note that LTEATR[V] bit has to be cleared to register subsequent errors in LTESR.

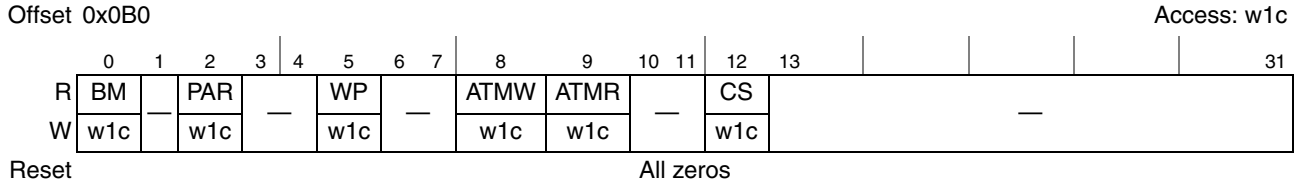


Figure 13-13. Transfer Error Status Register (LTESR)

Table 13-16 describes LTESR fields.

Table 13-16. LTESR Field Descriptions

Bits	Name	Description
0	BM	Bus monitor time-out 0 No local bus monitor time-out occurred. 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within LBCR[BMT] × 8 bus clock cycles from the start of a transaction.
1	—	Reserved
2	PAR	Parity 0 No local bus parity error 1 Local bus parity error. LTEATR[PB] indicates the byte lane that caused the error and LTEATR[BNK] indicates which memory controller bank was accessed.
3–4	—	Reserved
5	WP	Write protect error 0 No write protect error occurred. 1 A write was attempted to a local bus memory region that was defined as read-only in the memory controller. Usually, in this case, a bus monitor time-out will occur (as the cycle is not automatically terminated).
6–7	—	Reserved
8	ATMW	Atomic error write 0 No atomic write error occurred. 1 The subsequent write (WARA) to a memory bank did not occur within 256 bus clock cycles.

Table 13-16. LTESR Field Descriptions (continued)

Bits	Name	Description
9	ATMR	Atomic error read 0 No atomic read error occurred. 1 The subsequent read (RAWA) to a memory bank did not occur within 256 bus clock cycles.
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

13.3.1.11 Transfer Error Check Disable Register (LTEDR)

The transfer error check disable register (LTEDR), shown in Figure 13-14, 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.

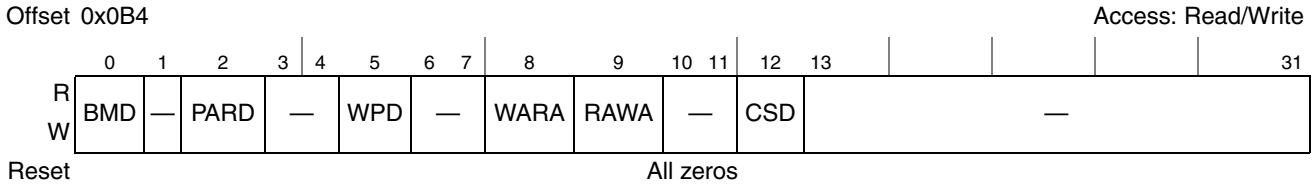


Figure 13-14. Transfer Error Check Disable Register (LTEDR)

Table 13-17 describes LTEDR fields.

Table 13-17. 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. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, PARD must be cleared and LTEIR[PARI] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 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

Table 13-17. LTEDR Field Descriptions (continued)

Bits	Name	Description
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
12	CSD	Chip select error checking disable 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–31	—	Reserved

13.3.1.12 Transfer Error Interrupt Enable Register (LTEIR)

The transfer error interrupt enable register (LTEIR), shown in Figure 13-15, is used to send or block error reporting through the LBC internal interrupt mechanism. Software should clear pending errors in LTESR before enabling interrupts. After an interrupt has occurred, clearing relevant LTESR error bits negates the interrupt.

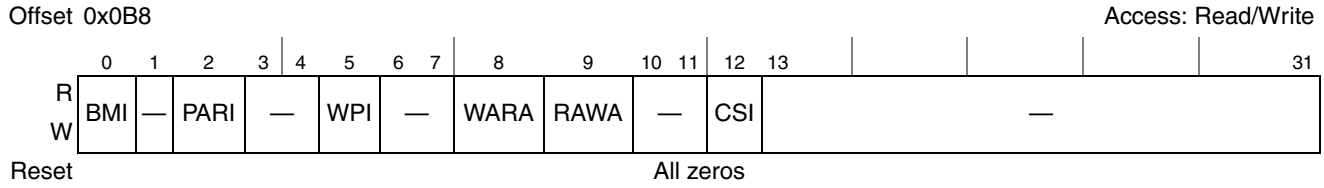


Figure 13-15. Transfer Error Interrupt Enable Register (LTEIR)

Table 13-18 describes LTEIR fields.

Table 13-18. LTEIR Field Descriptions

Bits	Name	Description
0	BMI	Bus monitor error interrupt enable 0 Bus monitor error reporting is disabled. 1 Bus monitor error reporting is enabled.
1	—	Reserved
2	PARI	Parity error interrupt enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, LTEDR[PARD] must be cleared and PARI must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 Parity error reporting is disabled. 1 Parity error reporting is enabled.
3–4	—	Reserved

Table 13-18. LTEIR Field Descriptions (continued)

Bits	Name	Description
5	WPI	Write protect error interrupt enable 0 Write protect error reporting is disabled. 1 Write protect error reporting is enabled.
6–7	—	Reserved
8	WARA	Write-after-read atomic (WARA) error interrupt enable 0 WARA error reporting is disabled. 1 WARA error reporting is enabled.
9	RAWA	Read-after-write atomic (RAWA) error interrupt enable 0 RAWA error reporting is disabled. 1 RAWA error reporting is enabled.
10–11	—	Reserved
12	CSI	Chip select error interrupt enable 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–31	—	Reserved

13.3.1.13 Transfer Error Attributes Register (LTEATR)

Figure 13-16 shows the LTEATR. After LTEATR[V] has been set, software must clear this bit to allow LBC error registers to update following any subsequent errors.

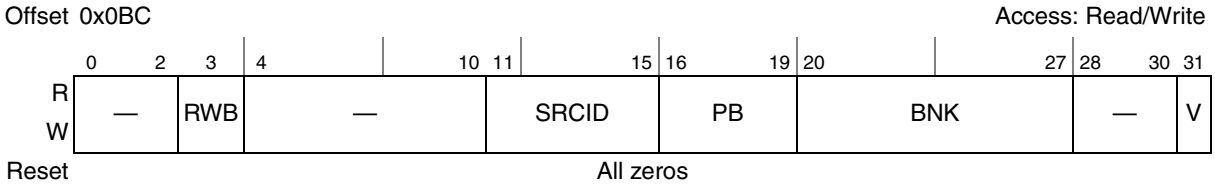


Figure 13-16. Transfer Error Attributes Register (LTEATR)

Table 13-19 describes LTEATR fields.

Table 13-19. 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.
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).

Table 13-19. LTEATR Field Descriptions (continued)

Bits	Name	Description
20–27	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.
28–30	—	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

13.3.1.14 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) is shown in Figure 13-17.

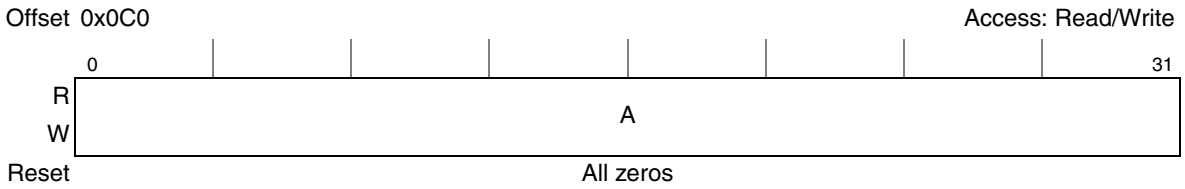


Figure 13-17. Transfer Error Address Register (LTEAR)

Table 13-20 describes LTEAR fields.

Table 13-20. LTEAR Field Descriptions

Bits	Name	Description
0–31	A	Transaction address for the error. Holds the 32-bit address of the transaction resulting in an error.

13.3.1.15 Local Bus Configuration Register (LBCR)

The local bus configuration register (LBCR) is shown in Figure 13-18.

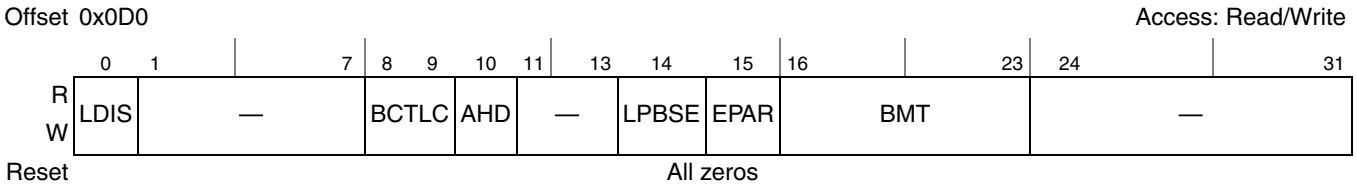


Figure 13-18. Local Bus Configuration Register

Table 13-21 describes LBCR fields.

Table 13-21. LBCR Field Descriptions

Bits	Name	Description
0	LDIS	Local bus disable 0 Local bus is enabled 1 Local bus is disabled. No internal transactions will be acknowledged.
1–7	—	Reserved
8–9	BCTLC	Defines the use of LBCTL 00 LBCTL is used as W/$\overline{R}$ control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as $\overline{LOE}$ for GPCM accesses only. 10 LBCTL is used as $\overline{LWE}$ for GPCM accesses only. 11 Reserved.
10	AHD	Address hold disable. Removes part of the hold time for LAD with respect to LALE in order to lengthen the LALE pulse 0 During address phases on the local bus, the LALE signal negates two platform clock periods prior to the address being invalidated. At 666 MHz, this provides 3 ns of additional address hold time at the external address latch. 1 During address phases on the local bus, the LALE signal negates one platform clock period prior to the address being invalidated. This halves the address hold time, but extends the latch enable duration. This may be necessary for very high frequency designs.
11–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal. 0 Parity byte select is disabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal is available for memory control as LGPL4 (output) or $\overline{LGTA}$/LUPWAIT (input). 1 Parity byte select is enabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal is dedicated as the parity byte select output, and $\overline{LGTA}$/LUPWAIT is disabled.
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.
24–31	—	Reserved

13.3.1.16 Clock Ratio Register (LCRR)

The clock ratio register sets the system (CCB) clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

Offset 0x0D4 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DBYP	—	BUFCMDC		—		ECL		—						EADC		—												CLKDIV			
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 13-19. Clock Ratio Register (LCRR)

Table 13-22 describes LCRR fields.

Table 13-22. LCRR Field Descriptions

Bits	Name	Description
0	DBYP	DLL bypass. This bit 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.It is recommended that DLL bypass mode be used at frequencies of 83 MHz or less. 0 The DLL is enabled. 1 The DLL is bypassed.
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
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. Note that LALE negates prior to the end of the final local bus clock, as controlled by LBCR[AHD]. 00 4 01 1 10 2 11 3

Table 13-22. LCRR Field Descriptions (continued)

Bits	Name	Description
16–27	—	Reserved
28–31	CLKDIV	System (CCB) clock divider. Sets the frequency ratio between the system (CCB) clock and the memory bus clock. Only the values shown in the table below are allowed. **Note:** It is critical that no transactions are being executed via the local bus while CLKDIV is being modified. As such, prior to modification, the user must ensure that code is not executing out of the local bus. Once LCRR[CLKDIV] is written, the register should be read, and then an isync should be executed. 0000–0001 Reserved 0010 2 0011 Reserved 0100 4 0101–0111 Reserved 1000 8 1001–1111 Reserved

13.4 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.

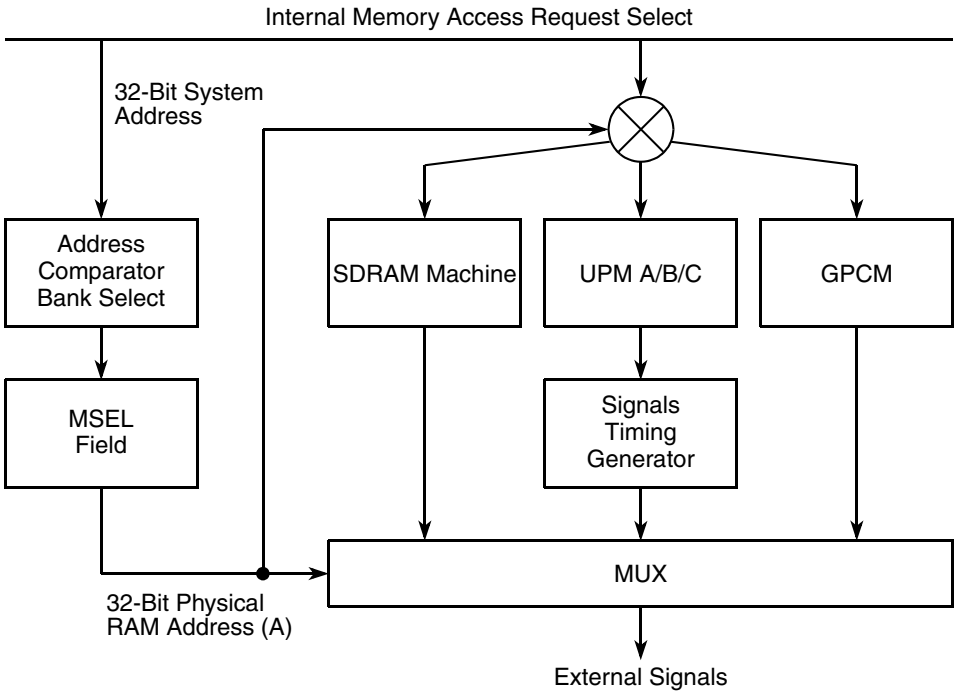


Figure 13-20. Basic Operation of Memory Controllers in the LBC

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 13-20. 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.

13.4.1 Basic Architecture

The following sections describe the basic architecture of the LBC.

13.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BR*n* registers, while 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).

13.4.1.2 External Address Latch Enable Signal (LALE)

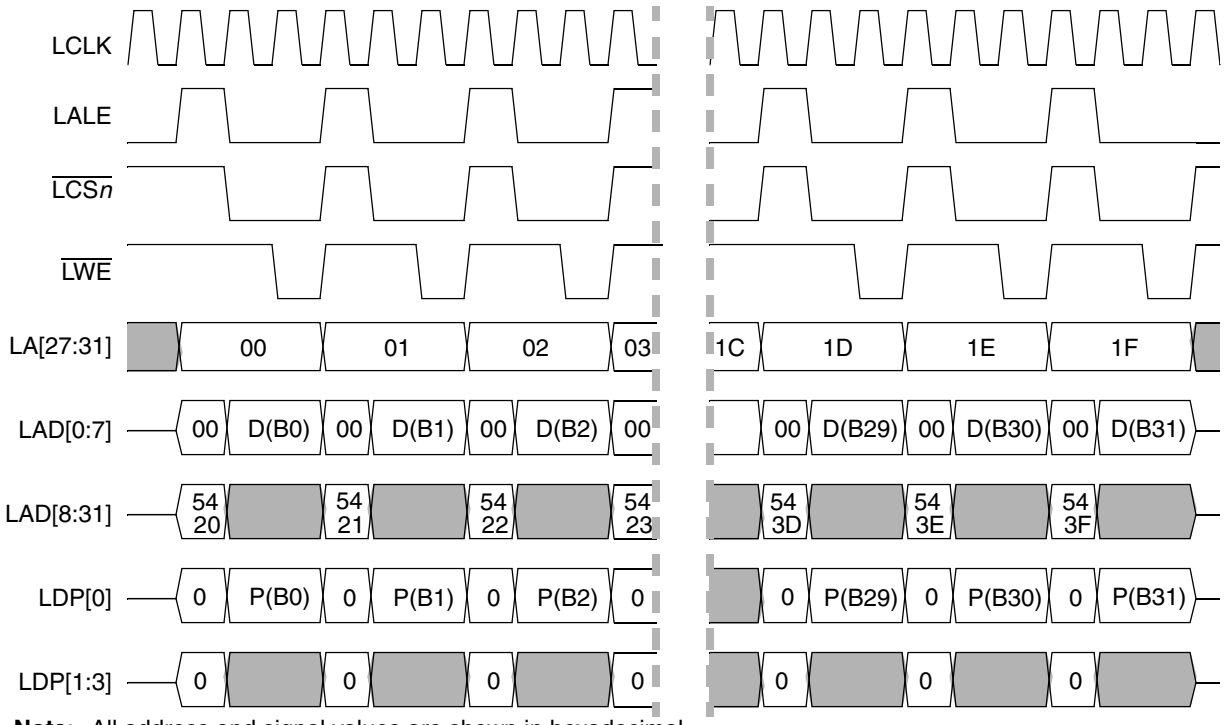
The local bus uses a multiplexed address/data bus. Therefore, the LBC must distinguish between address and data phases, which take place on the same bus (LAD[0:31] signals). The LALE signal, when asserted, signifies an address phase during which the LBC drives the memory address on the LAD[0:31] signals.

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 (bidirectional) 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.

To ensure adequate hold time on the external address latch, LALE negates earlier than the address changes on LAD[0:31] during address phases. By default, LALE negates earlier by two platform clock periods (which, divided by LCRR[CLKDIV], yields the local bus clock). For example, if the LBC is operating at 666 MHz internally, then an additional 3 ns of address hold time is introduced. However, when LCRR[CLKDIV] = 2 and the LCLK frequency exceeds 100 MHz, the duration of the shortened LALE pulse may not meet the minimum latch enable pulse width specifications of some latches. In such cases, setting LBCR[AHD] = 1 increases the LALE pulse width by one platform clock cycle, and decreases the address hold time by the same amount. At 666 MHz and with LCRR[CLKDIV] = 2, the duration of LALE would then be 4.5 ns, with 1.5 ns of hold time. If both longer hold time and longer LALE pulse duration are needed, then the address phase can be extended using the ORn[EAD] and LCRR[EADC] fields, and the LBCR[AHD] bit can be left at 0. However, this will add latency to all address tenures.

The frequency of LALE assertion varies across the three memory controllers. For GPCM, every assertion of $\overline{\text{LCS}n}$ 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 $\overline{\text{LCS}n}$ 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 through 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, on 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 13-21 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 LDP[0] are driven with valid data and parity, respectively.



Note: All address and signal values are shown in hexadecimal.
D(Bk) = kth of 32 data bytes, P(Bk) = parity bit of kth data byte.

Figure 13-21. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420

13.4.1.3 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 MDVAL 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 13-22 shows LALE, TA (internal), and $\overline{\text{LCS}n}$. Note that TA and LALE are never asserted together, and that for the duration of LALE, $\overline{\text{LCS}n}$ (or any other control signal) remains negated or frozen.

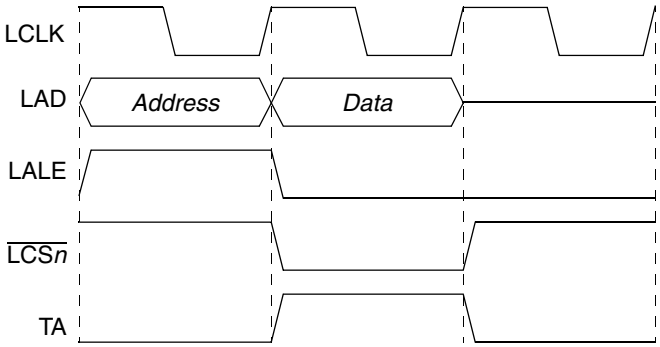


Figure 13-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}n}$

13.4.1.4 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 OR*n*[BCTLD]. Access to an SDRAM machine controlled bank does not activate the LBCTL control. LBCTL can be further configured by LBCR[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 (LBCR[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.

13.4.1.5 Atomic Operation

The LBC supports the following kinds of atomic bus operations (set by BR*n*[ATOM]):

- Read-after-write atomic (RAWA). When a write access hits a memory bank in which 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).

13.4.1.6 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming BR*n*[DECC]. Parity is generated and checked on a per-byte basis using LDP[0:3] for the bank if BR*n*[DECC] = 01 (normal parity) or BR*n*[DECC] = 10 for read-modify-write (RMW) parity. Byte lane parity on LDP[0:3] is generated regardless of the BR*n*[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).

13.4.1.7 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 13.4.4.1.4, "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.

13.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPROM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups—BR*n* and OR*n*.

Figure 13-23 shows a simple connection between an 8-bit port size SRAM device and the LBC in GPCM mode. Byte-write enable signals ($\overline{\text{LWE}}$) are available for each byte written to memory. Also, the output enable signal ($\overline{\text{LOE}}$) is provided to minimize external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select ($\overline{\text{LCS0}}$) prior to the system being fully configured.

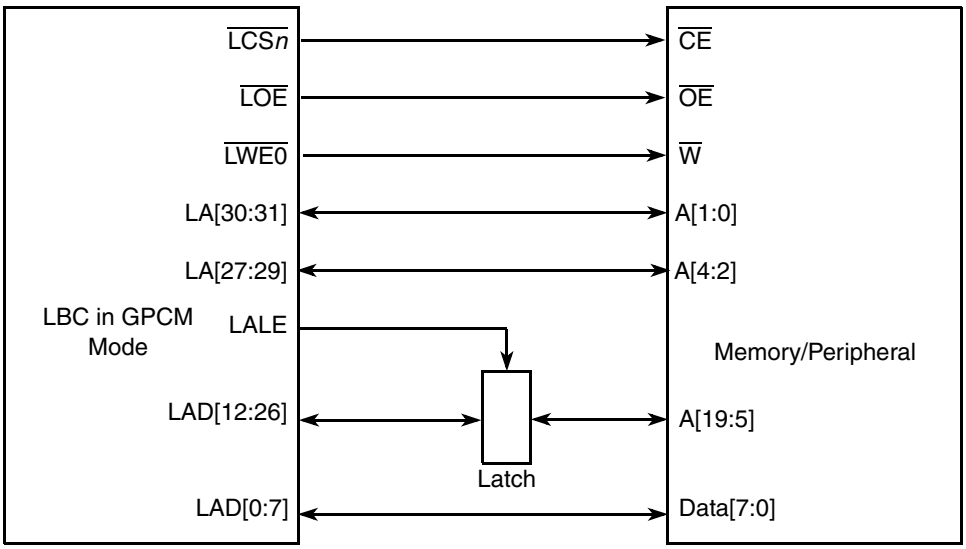


Figure 13-23. Local Bus to GPCM Device Interface

Figure 13-24 shows $\overline{\text{LCS}}$ as defined by the setup time required between the address lines and $\overline{\text{LCS}}$. The user can configure OR*n*[ACS] to specify $\overline{\text{LCS}}$ to meet this requirement.

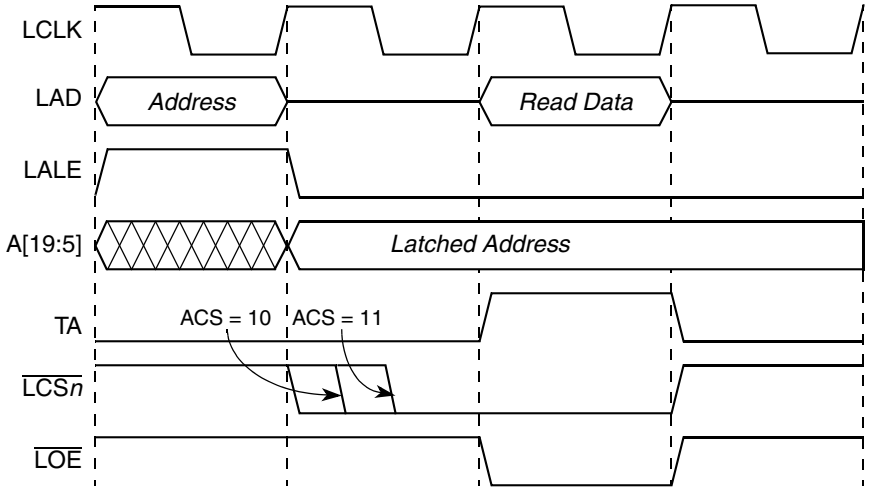


Figure 13-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4, 8)

13.4.2.1 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, TRLX, EHTR, and SETA fields.

Table 13-23 shows signal behavior and system response for a write access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8.

Table 13-23. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}}$*n* Asserted	$\overline{\text{LCS}}$*n* 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	1/4	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

Table 13-23. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 4 or 8 (continued)

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ Negated to Address Change	$\overline{\text{LWE}}$ Negated to Address/Data Invalid	Total Cycles[1]
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 13-24 shows the signal behavior and system response for a read access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8.

Table 13-24. 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

Table 13-24. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8 (continued)

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]
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 13-25 and Table 13-26 show the write and read signal behavior, respectively, when LCRR[CLKDIV] = 2.

Table 13-25. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ 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

Table 13-25. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2 (continued)

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ Negated to Address Change	$\overline{\text{LWE}}$ Negated to Address/Data Invalid	Total Cycles[1]
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*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 13-26. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock Cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ Negated to Address Change	Total Cycles[1]
0	0	0	00	0	1	4+SCY
0	0	0	10	1/2	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

Table 13-26. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2 (continued)

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]
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 1 cycle only (OR*n*[EAD] = 0; OR*n*[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than 1 cycle increases the total cycle count accordingly.

13.4.2.2 Chip-Select Assertion Timing

The banks selected to work with the GPCM support an option to drive the $\overline{\text{LCS}n}$ signal with different timings (with respect to the external address/data bus). $\overline{\text{LCS}n}$ can be driven in any of the following ways:

- Simultaneous with the latched memory address. (This refers to the externally latched address, not the address timing on LAD[0:31]. That is, chip select does not assert during LALE).
- One quarter of a clock cycle later (for LCRR[CLKDIV] = 4 or 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.

The timing diagram in Figure 13-24 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.

13.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between zero 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.

13.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 13-23 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. The $\overline{\text{LWE}}$[0:3] signals are connected to the respective $\overline{\text{WE}}$[3:0] signals on the memory device where each $\overline{\text{LWE}}$[0:3] signal corresponds to a different data byte.

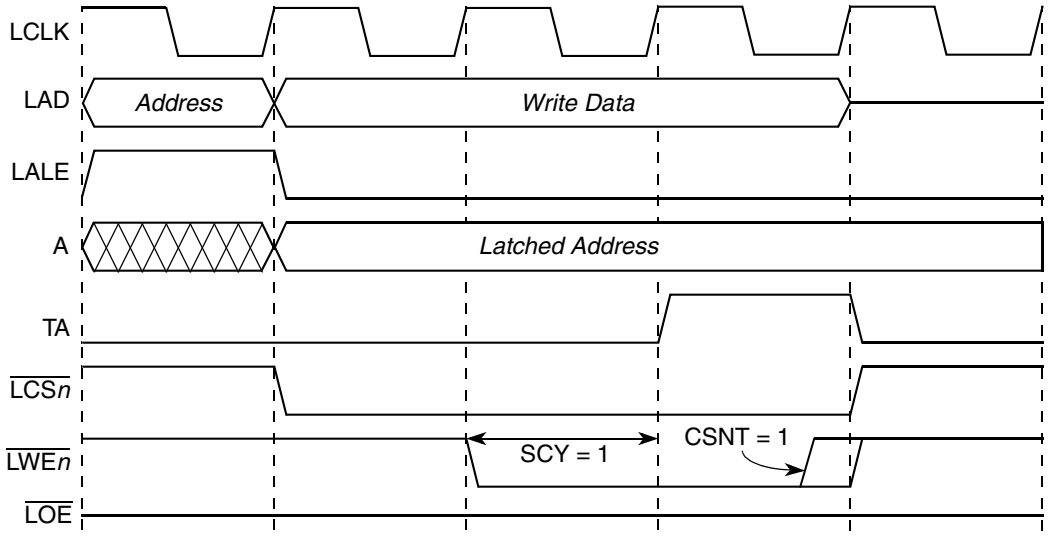


Figure 13-25. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4 or 8)

As Figure 13-25 shows, the timing for $\overline{\text{LCS}n}$ is the same as for the latched address. The strobes for the transaction are supplied by $\overline{\text{LOE}}$ or $\overline{\text{LWE}n}$, depending on the transaction direction—read or write (write case shown in Figure 13-25). 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[CLKDIV] = 4 or 8. For example, when ACS = 00 and CSNT = 1, $\overline{\text{LWE}n}$ is negated one quarter of a clock earlier, as shown in Figure 13-25. If LCRR[CLKDIV] = 2, $\overline{\text{LWE}n}$ is negated either coincident with $\overline{\text{LCS}n}$ or one cycle earlier.

13.4.2.2.3 Relaxed Timing

OR*x*[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 ≠ 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.
- $\overline{\text{LCS}n}$ signals are negated 1 cycle earlier during writes (but only if ACS ≠ 00).
- $\overline{\text{LWE}}$[0:3] signals are negated 1 cycle earlier during writes.

Figure 13-26 and Figure 13-27 show relaxed timing read and write transactions. The effect of CLKDIV = 2 for these examples is only to delay the assertion of $\overline{\text{LCS}n}$ in the ACS = 10 case to the ACS = 11 case. The example in Figure 13-27 also shows address and data multiplexing on LAD[0:31] for a pair of writes issued consecutively.

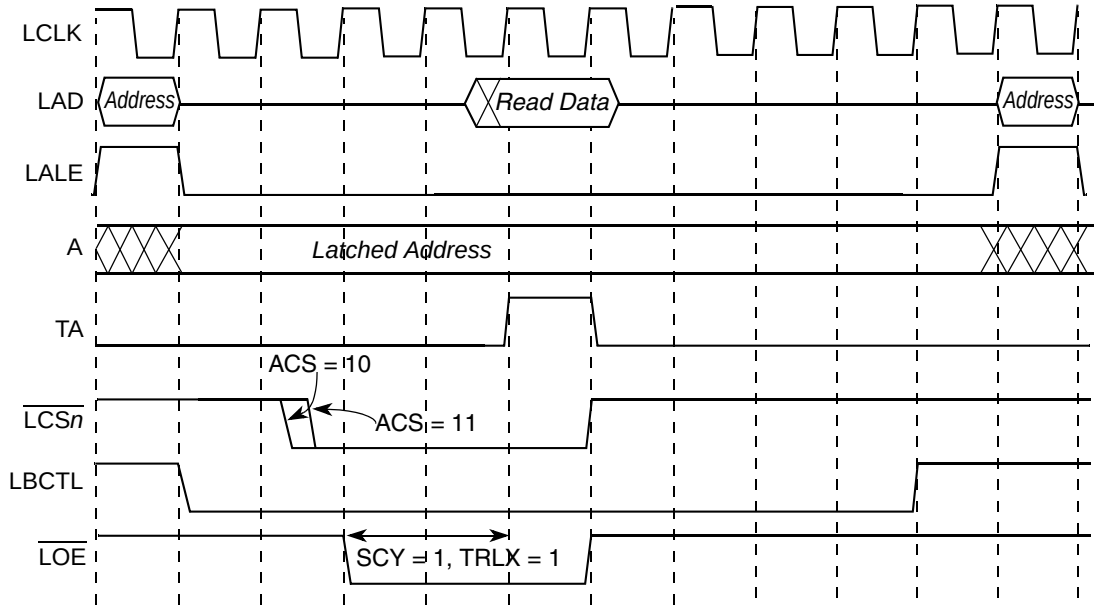


Figure 13-26. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, EHTR = 0, TRLX = 1)

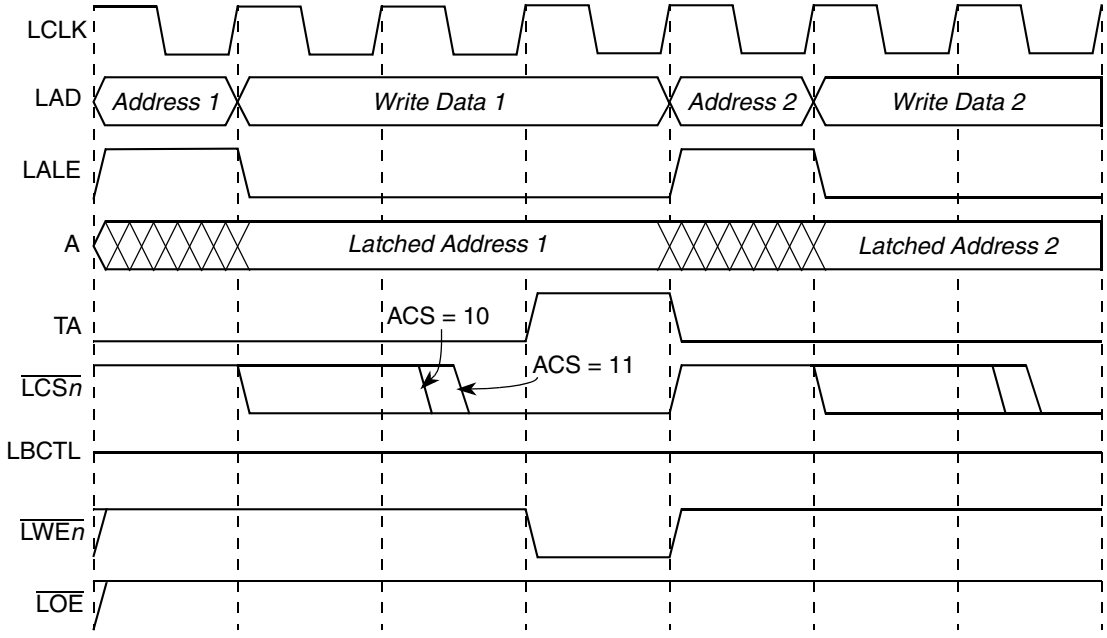


Figure 13-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4 or 8)

When TRLX and CSNT are set in a write access, the $\overline{\text{LWE}}$[0:3] strobe signals are negated one clock earlier than in the normal case, as shown in Figure 13-28 and Figure 13-29. If ACS ≠ 00, $\overline{\text{LCS}n}$ is also negated one clock earlier.

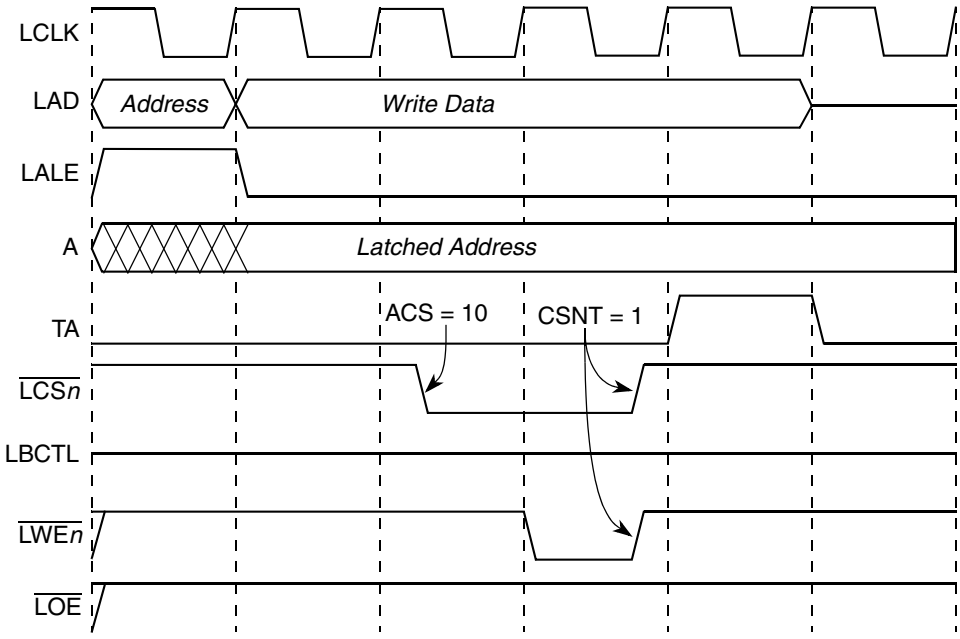


Figure 13-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4 or 8)

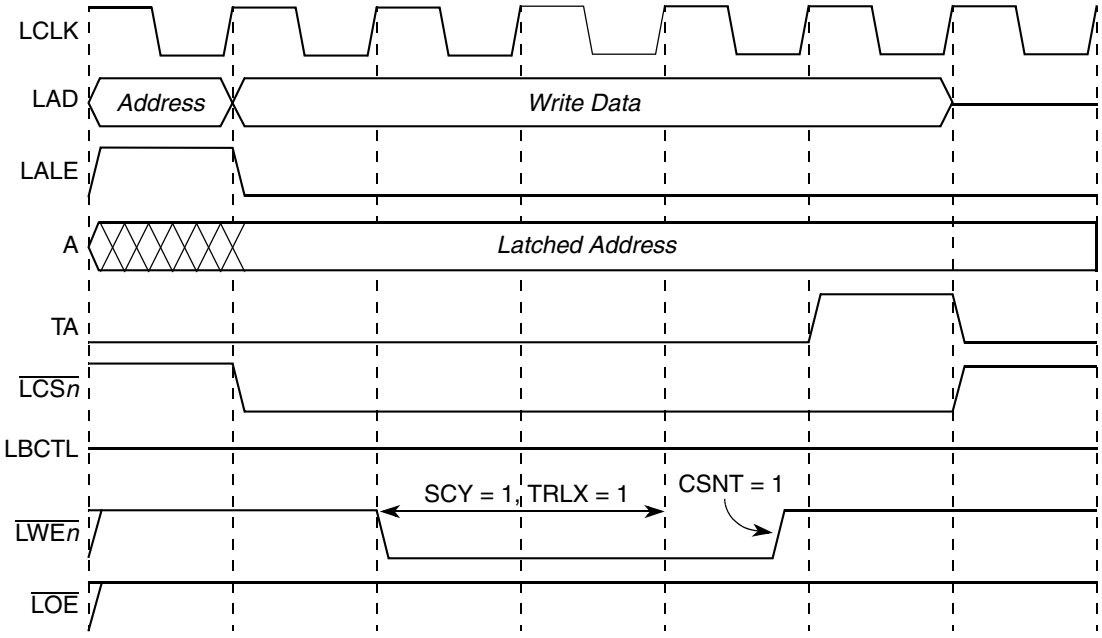


Figure 13-29. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4 or 8)

13.4.2.2.4 Output Enable ($\overline{\text{LOE}}$) Timing

The timing of $\overline{\text{LOE}}$ is affected only by TRLX. It always asserts and negates on the rising edge of the bus clock. $\overline{\text{LOE}}$ asserts either on the rising edge of the bus clock after $\overline{\text{LCS}}n$ is asserted or coinciding with $\overline{\text{LCS}}n$ (if XACS = 1 and ACS = 10 or 11). Accordingly, assertion of $\overline{\text{LOE}}$ can be delayed (along with the

assertion of $\overline{\text{LCS}n}$) by programming TRLX = 1. $\overline{\text{LOE}}$ negates on the rising clock edge coinciding with $\overline{\text{LCS}n}$ negation

13.4.2.2.5 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 OR*n*[TRLX,EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified by the configuration of OR*n*[TRLX,EHTR], as described in Section 13.3.1.2.2, "Option Registers (ORn)—GPCM Mode," 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 OR*n*[EHTR]. Figure 13-30, and Figure 13-31 present various GPCM timing examples.

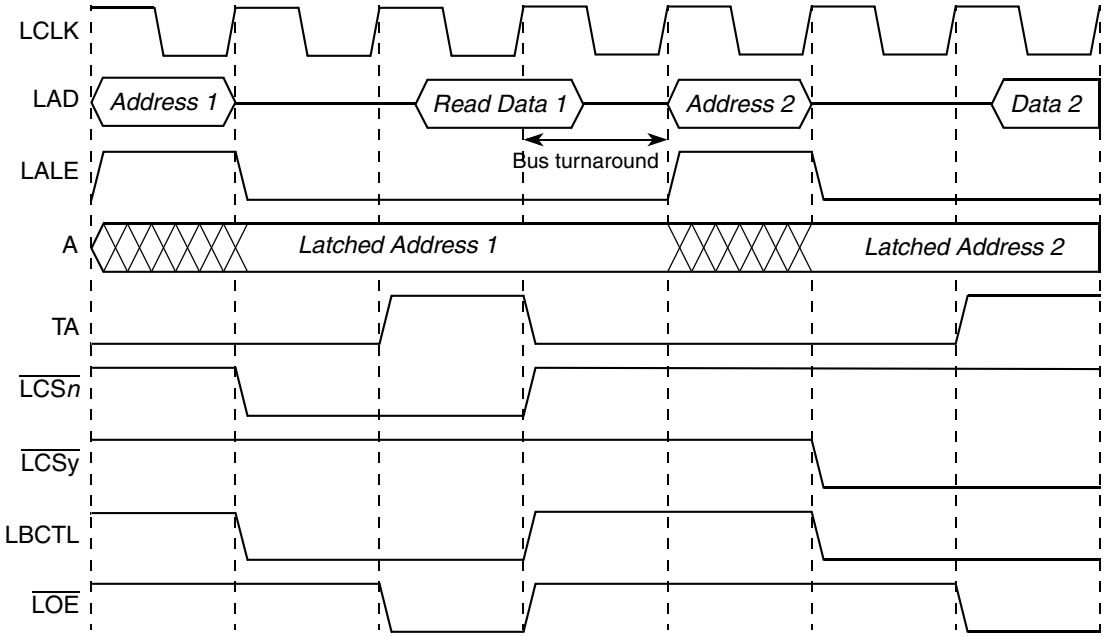


Figure 13-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

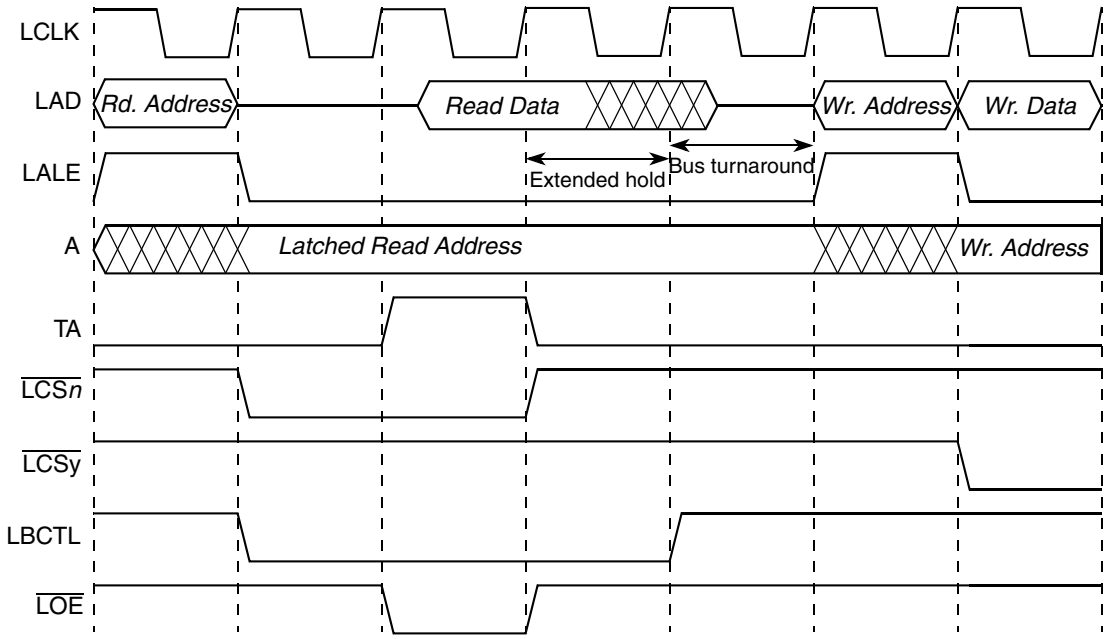


Figure 13-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-Cycle Extended Hold Time on Reads)

13.4.2.3 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 $\overline{\text{LCS}}n$, 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 OR*n*[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 the case of a 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 OR*n*[SETA]. Asserting $\overline{\text{LGTA}}$ always terminates an access, even if OR*n*[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 $\overline{\text{LGTA}}$ is illustrated by the example in Figure 13-32.

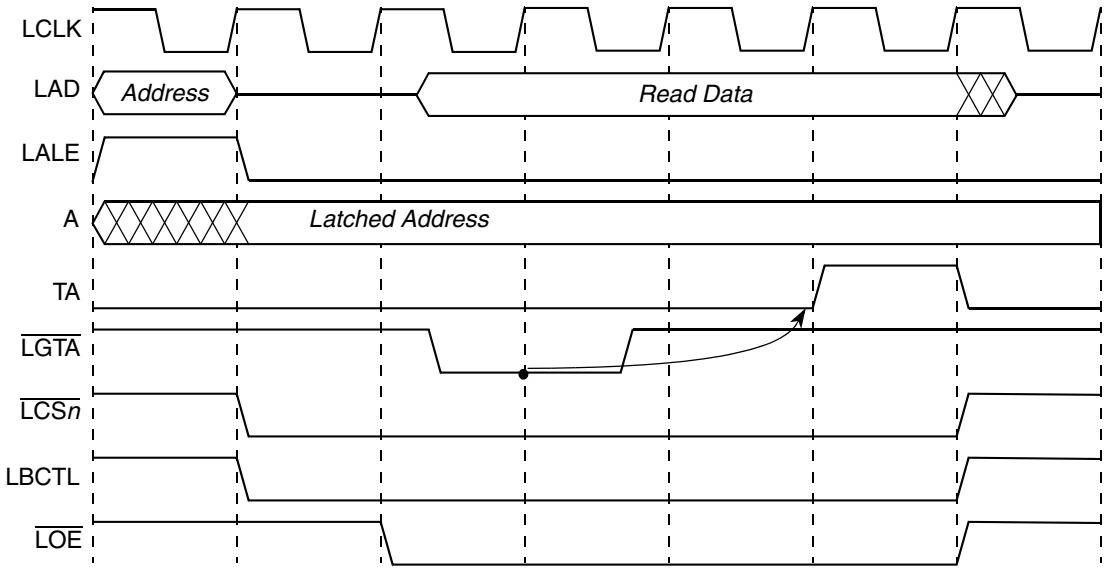


Figure 13-32. External Termination of GPCM Access

13.4.2.4 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 13-27 describes the initial values of the boot bank in the memory controller.

Table 13-27. Boot Bank Field Values After Reset

Register	Field	Setting	Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0	OR0	AM	0000_0000_0000_0000_0
	PS	From signal during reset.		BCTLD	0
	DECC	00		CSNT	1
	WP	0		ACS	11
	MSEL	000		XACS	1
	ATOM	00		SCY	1111
	V	1		SETA	0
				TRLX	1
				EHTR	1
				EAD	1

13.4.3 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.

13.4.3.1 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 13-33 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].

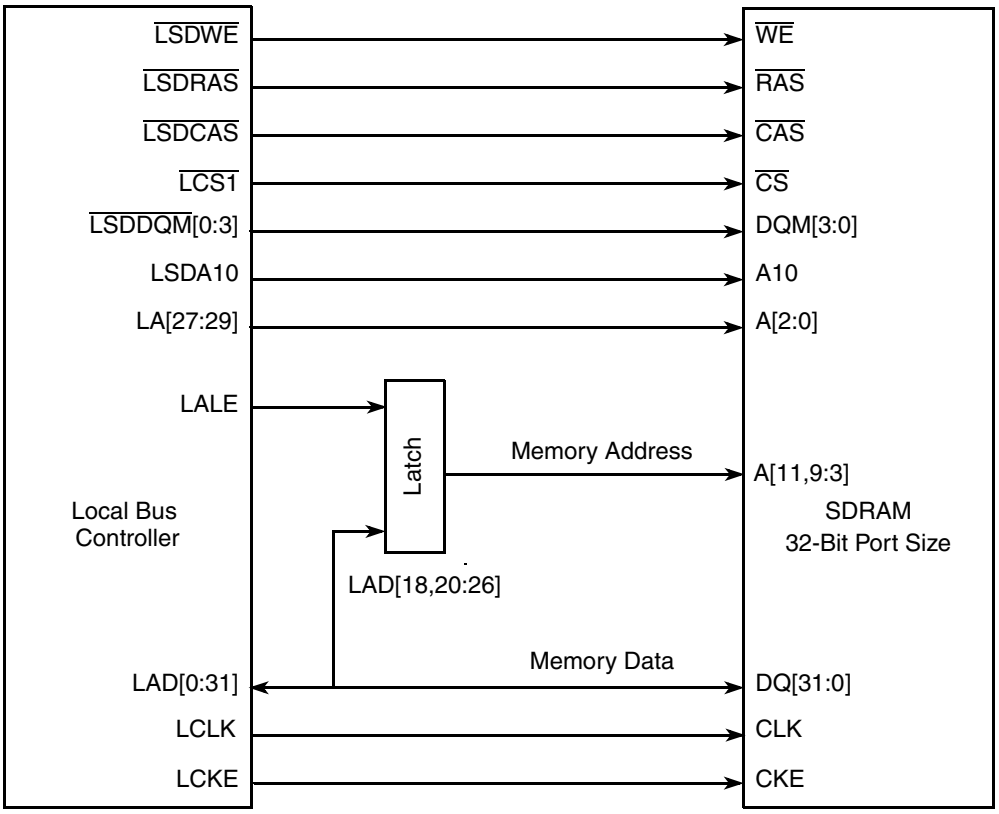


Figure 13-33. Connection to a 32-Bit SDRAM with 12 Address Lines

13.4.3.2 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 e500, 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.

13.4.3.3 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 13-28. 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.

Table 13-28. SDRAM Interface Commands (continued)

Command (LSDMR[OP])	Description
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. $\overline{\text{LSDDQM}}$[0:3] are inactive and write data is undefined.
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 when the rest of the LBC is in a power saving mode with clocks turned off. When placed in this mode, the SDRAM device is capable of issuing 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 minimum of 200 bus cycles before issuing reads or writes to the LBC.

13.4.3.4 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.

13.4.3.5 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:

- 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.

13.4.3.6 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 13-34 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 13-34) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).

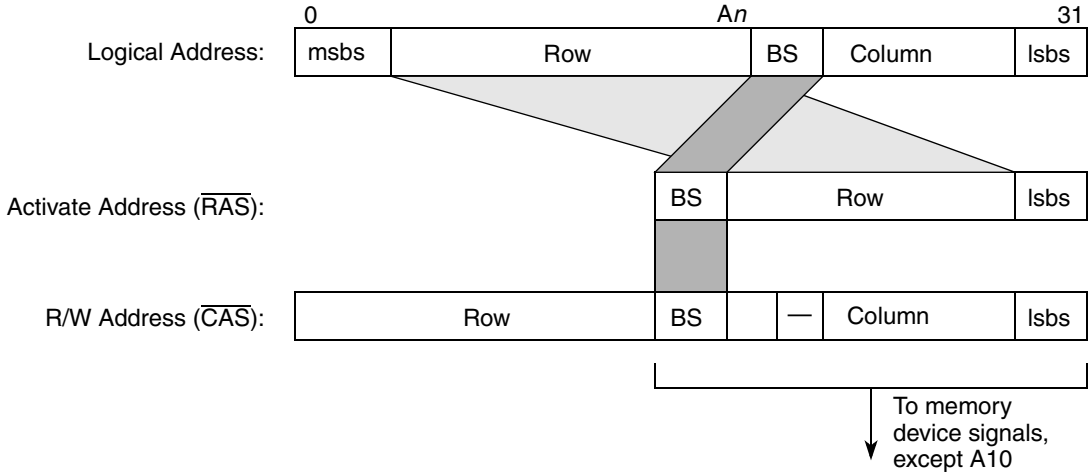


Figure 13-34. SDRAM Address Multiplexing

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.

13.4.3.7 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.

13.4.3.7.1 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.

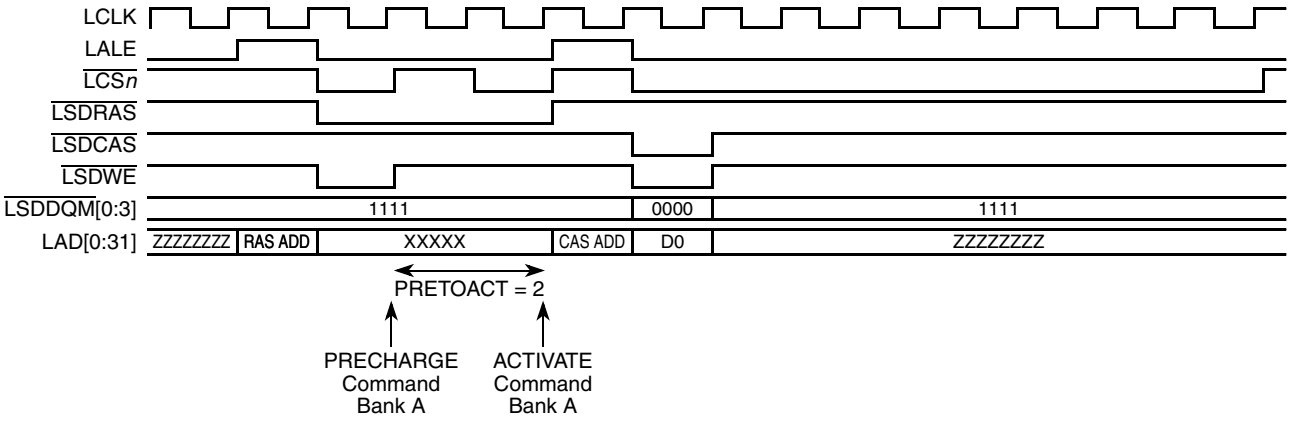


Figure 13-35. PRETOACT = 2 (2 Clock Cycles)

13.4.3.7.2 Activate-to-Read/Write Interval

This parameter, controlled by LSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command to the same SDRAM bank.

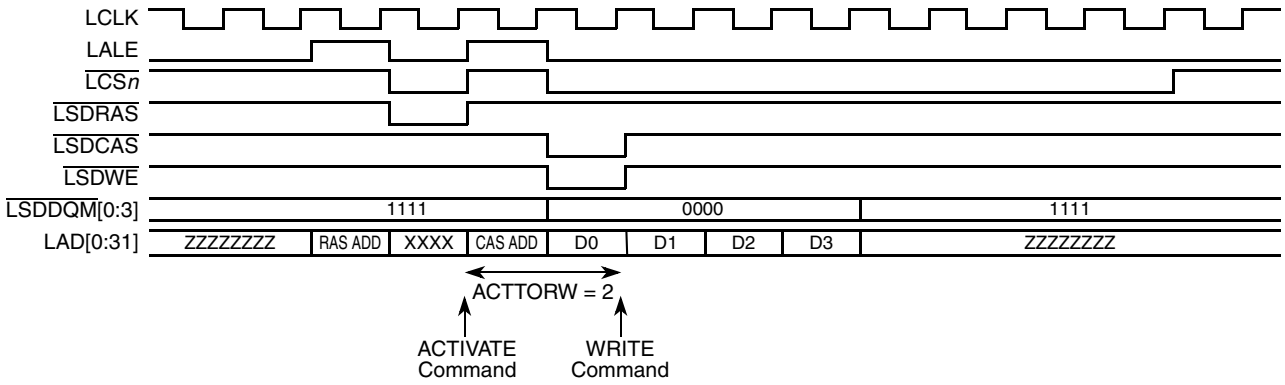


Figure 13-36. ACTTORW = 2 (2 Clock Cycles)

13.4.3.7.3 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.

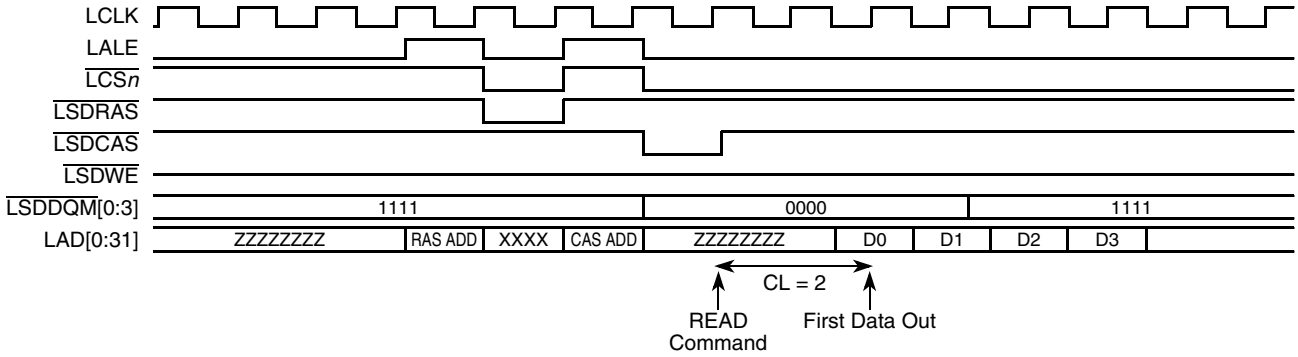


Figure 13-37. CL = 2 (2 Clock Cycles)

13.4.3.7.4 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.

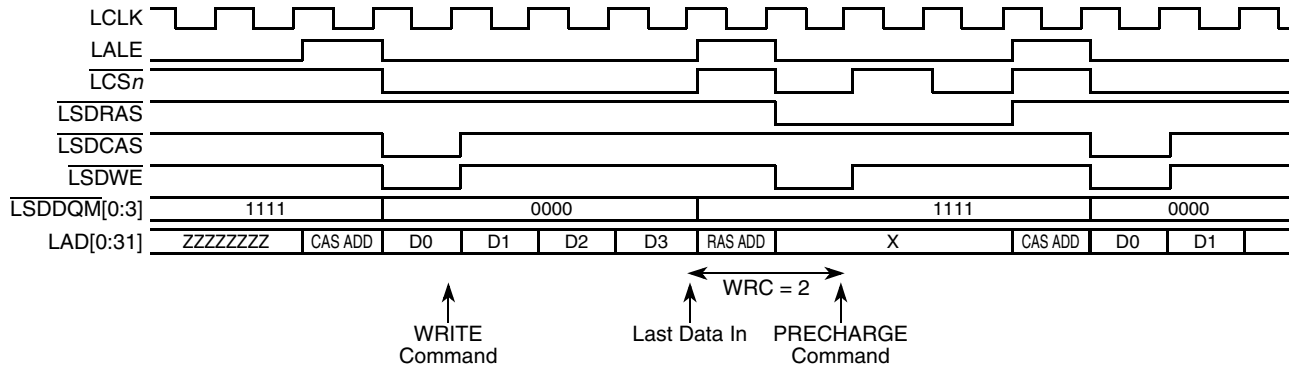


Figure 13-38. WRC = 2 (2 Clock Cycles)

13.4.3.7.5 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.

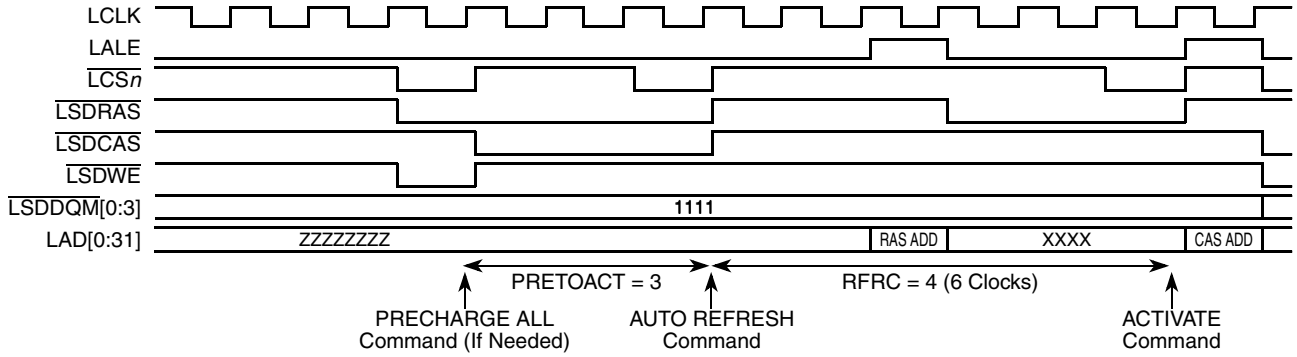


Figure 13-39. RFRC = 4 (6 Clock Cycles)

13.4.3.7.6 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 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 LSDA10) for each SDRAM command.

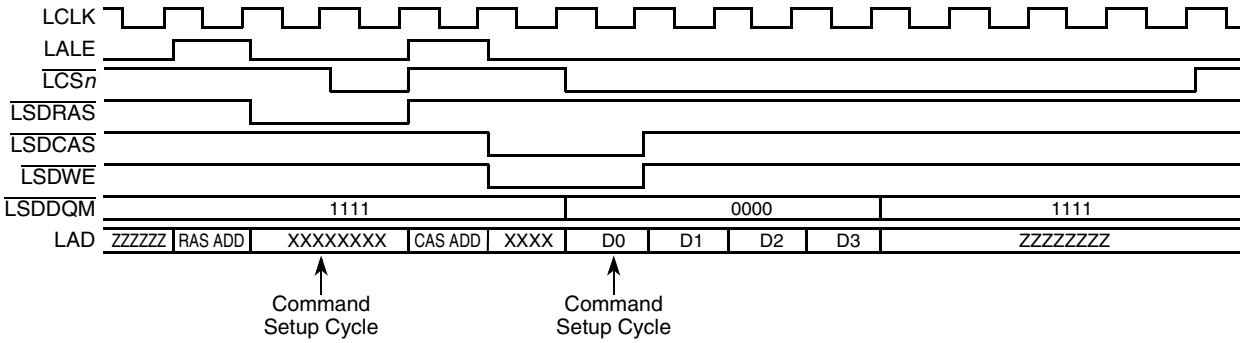


Figure 13-40. BUFCMD = 1, LCRR[BUFCMDC] = 2

13.4.3.8 SDRAM Interface Timing

The following figures show SDRAM timing for various types of accesses.

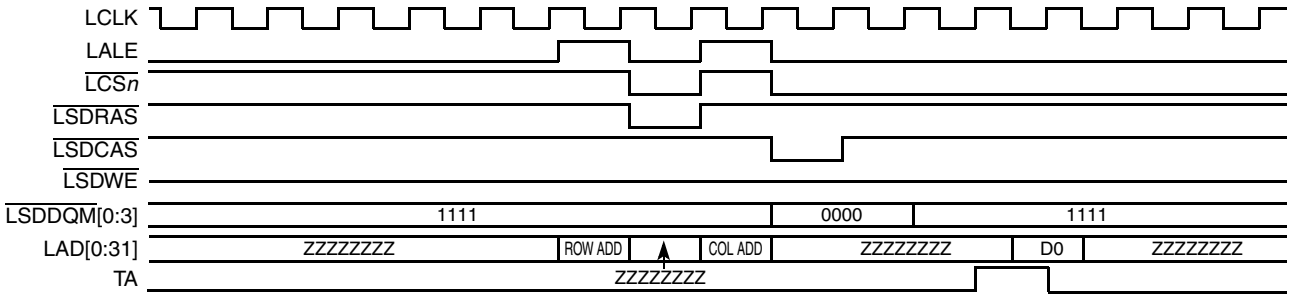


Figure 13-41. SDRAM Single-Beat Read, Page Closed, CL = 3

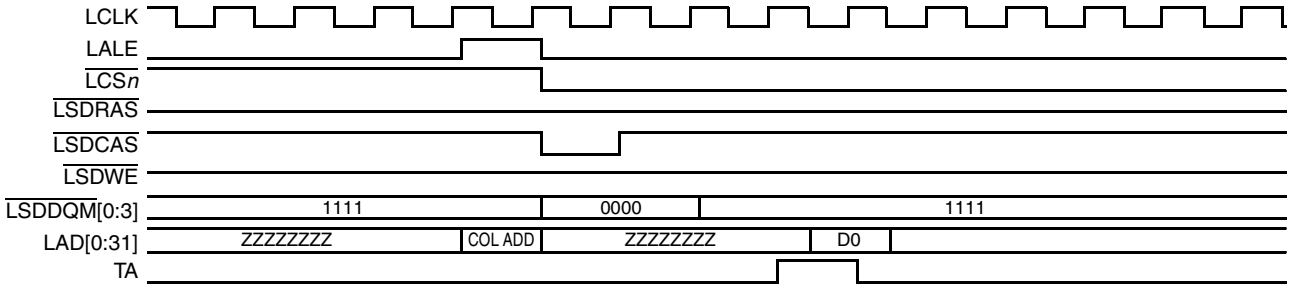


Figure 13-42. SDRAM Single-Beat Read, Page Hit, CL = 3

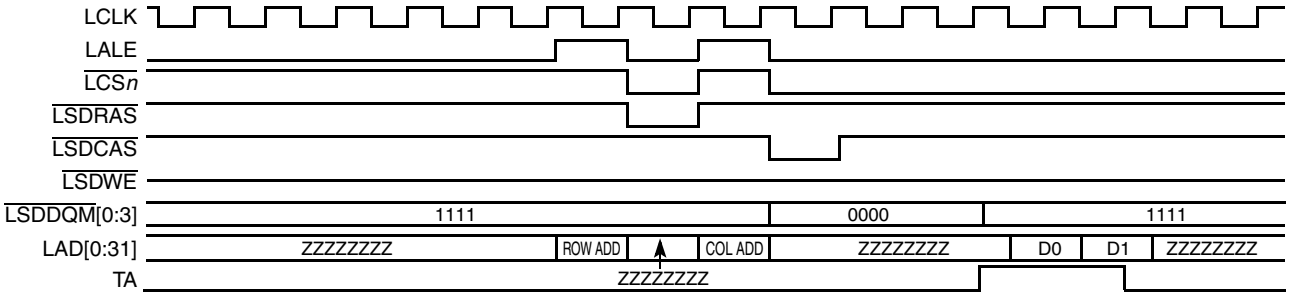


Figure 13-43. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

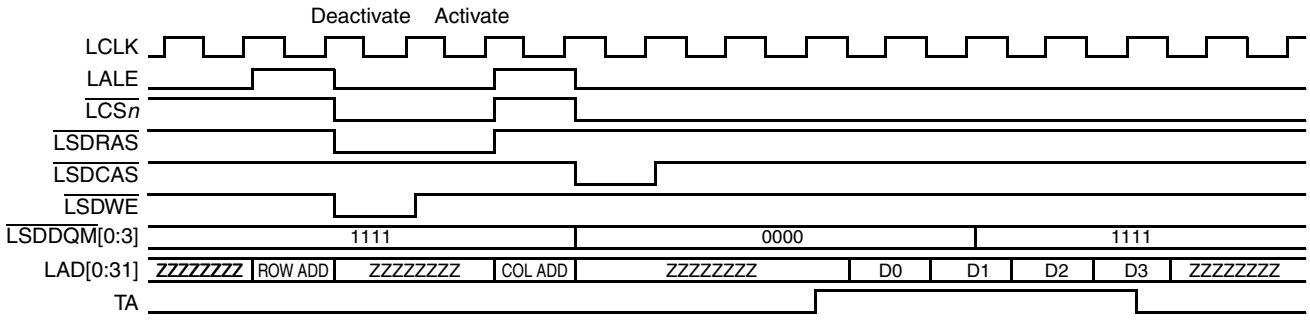


Figure 13-44. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

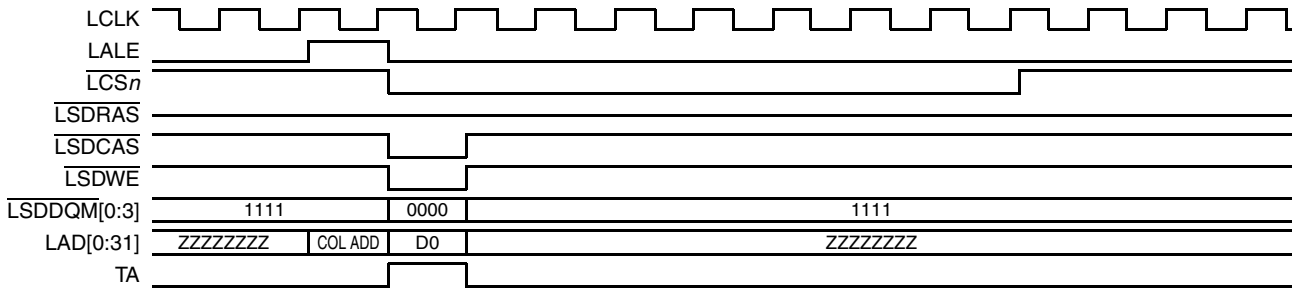


Figure 13-45. SDRAM Single-Beat Write, Page Hit

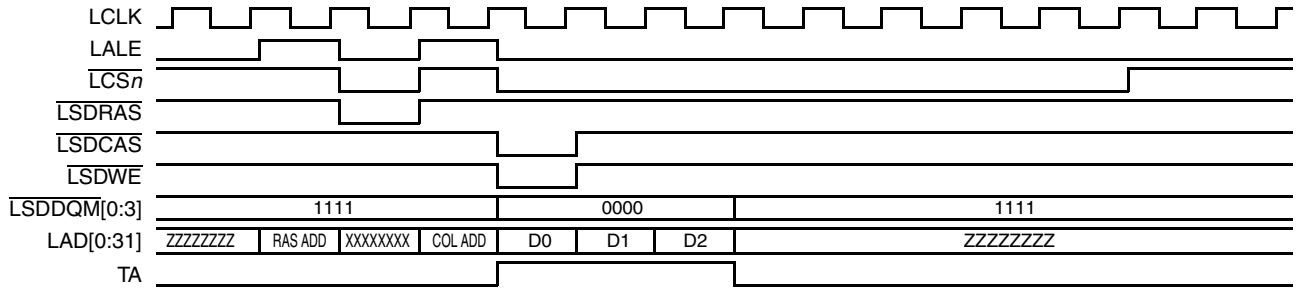


Figure 13-46. SDRAM Three-Beat Write, Page Closed

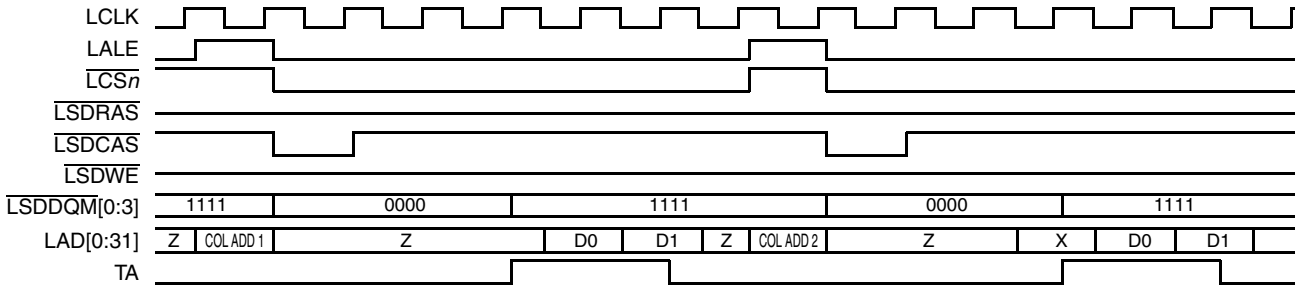


Figure 13-47. SDRAM Read-After-Read Pipelined, Page Hit, CL = 3

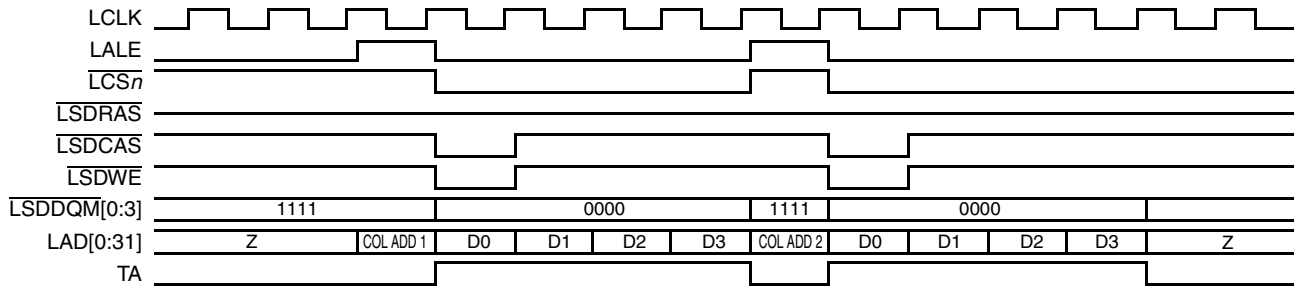


Figure 13-48. SDRAM Write-After-Write Pipelined, Page Hit

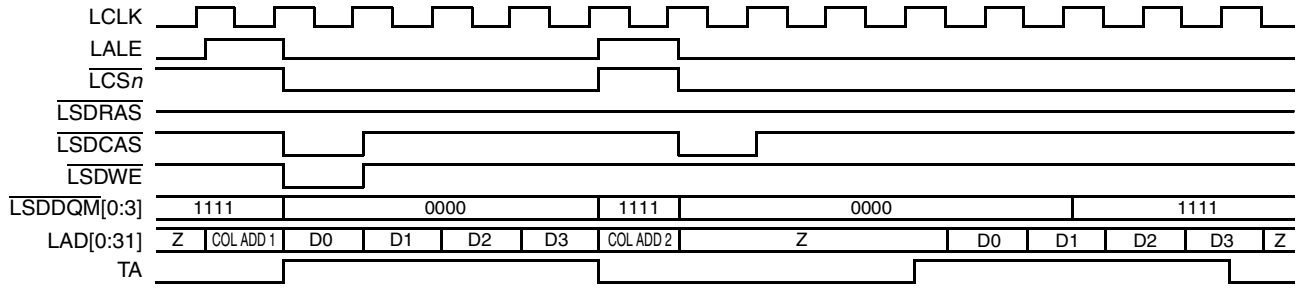


Figure 13-49. SDRAM Read-After-Write Pipelined, Page Hit

13.4.3.9 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-bit 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.

13.4.3.10 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 13-50. In this case, the latched address carries the mode bits for the command.

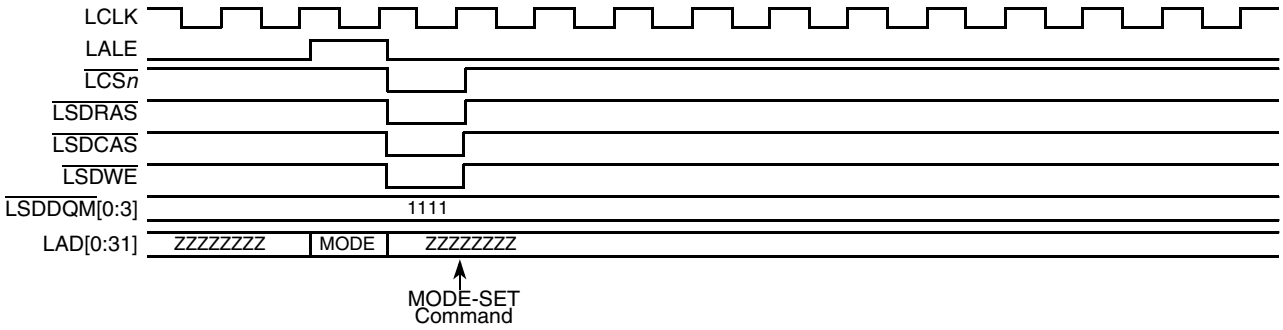


Figure 13-50. SDRAM MODE-SET Command

13.4.3.11 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 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 period of the refresh interval 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.

13.4.3.11.1 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 S

DRAM machine's mode register (LSDMR[RFCR]) 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[RFCR].

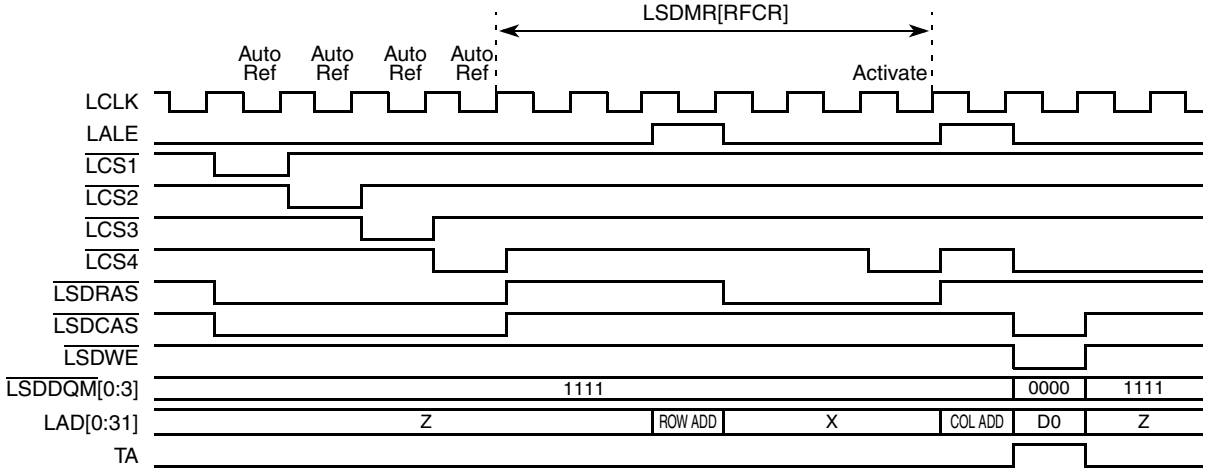


Figure 13-51. SDRAM Bank-Staggered Auto-Refresh Timing

13.4.4 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 ($\overline{\text{LCS}}n$, $\overline{\text{LBS}}$[0:3], and 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.

NOTE

If the LGPL4/$\overline{\text{LGTA}}$/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pull-up is required. Refer to the hardware specification for details regarding termination options.

Figure 13-52 shows the basic operation of each UPM.

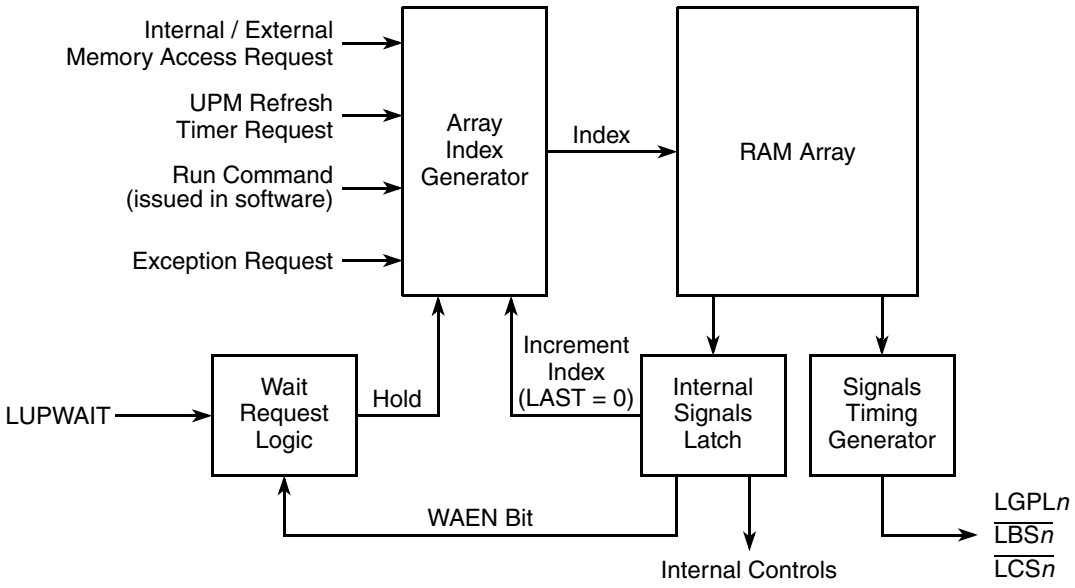


Figure 13-52. User-Programmable Machine Functional Block Diagram

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.

13.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each of the possible UPM requests. An internal device's request for a memory access initiates one of the following patterns (M*x*MR[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 13-53 and Table 13-29 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (M*x*MR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

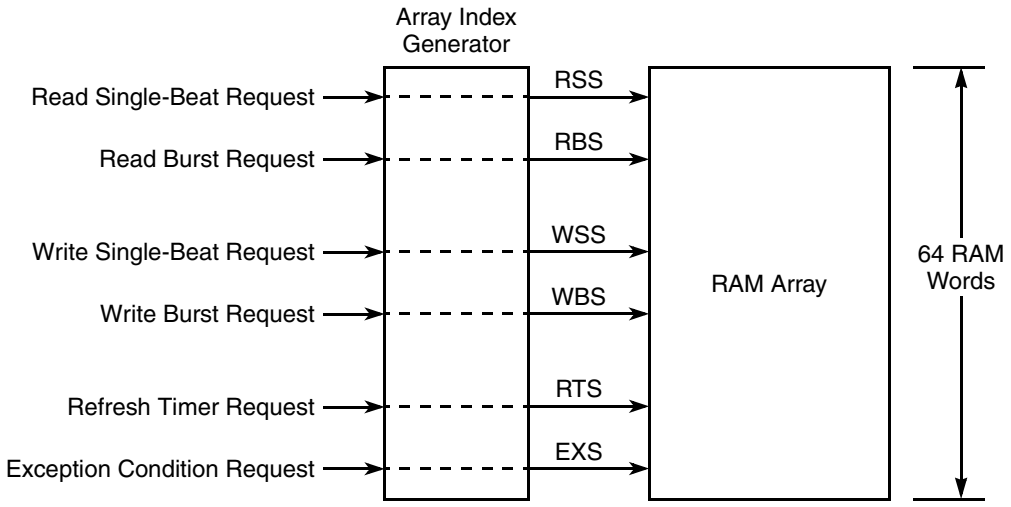


Figure 13-53. RAM Array Indexing

Table 13-29. 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

13.4.4.1.1 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 4 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.

13.4.4.1.2 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 13-54 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.

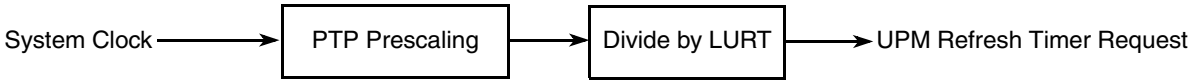


Figure 13-54. Memory Refresh Timer Request Block Diagram

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*x*MR[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.

13.4.4.1.3 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*x*MR[OP] = 11 and accessing UPM*n* memory region with any write transaction that hits the corresponding UPM machine. M*x*MR[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.

13.4.4.1.4 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.

13.4.4.2 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 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*x*MR.

Patterns are written to the RAM array by setting M*x*MR[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*x*MR[OP] = 10).

NOTE

M*x*MR/MDR registers should not be updated while dummy read/write accesses are still in progress. Dummy transaction completion is indicated by incremented M*x*MR[MAD]. In order to enforce proper ordering between updates to the M*x*MR register and the dummy accesses to the UPM memory region, two rules must be followed:

1.) Since the result of any update to the M*x*MR/MDR register must be in effect before the dummy read or write to the UPM region, a write to M*x*MR/MDR should be followed immediately by a read of M*x*MR/MDR.

2.) The UPM memory region should have the same MMU settings as the memory region containing the M*x*MR configuration register; both should be mapped by the MMU as cache-inhibited and guarded. This prevents the core from re-ordering a read of the UPM memory around the read of M*x*MR. Once 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.

13.4.4.2.1 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*n* and OR*n* registers have been previously setup.

1. Program M*x*MR for the first write (with desired RAM array address).
2. Write pattern/data to MDR to ensure that the M*x*MR 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*x*MR if step 2 is not performed.)
4. Preform a dummy write transaction. (Write transaction can now be performed.)
5. Read/check M*x*MR[MAD]. If incremented, then the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program M*x*MR for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the M*x*MR 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*x*MR[MAD]. If incremented, then the previous dummy write transaction is completed.

Note that if step 1 (or 6) and 2 (or 7) are reversed, then step 3 (or 8) is replaced by the following:

- Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration.

13.4.4.2.2 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 (when M*x*MR[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*n* and OR*n* registers have been previously setup.

1. Program M*x*MR for the first read with the desired RAM array address.
2. Read M*x*MR to ensure that the M*x*MR 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*x*MR[MAD]. If incremented, then the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program M*x*MR for the second read with the desired RAM array address.
7. Read M*x*MR to ensure that the M*x*MR 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*x*MR[MAD]. If incremented, then the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

13.4.4.3 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 $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$ 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 13-55 and Figure 13-56. 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.

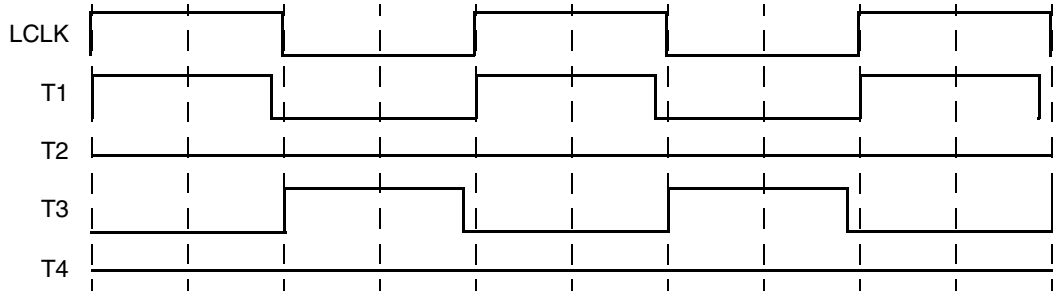


Figure 13-55. UPM Clock Scheme for LCRR[CLKDIV] = 2

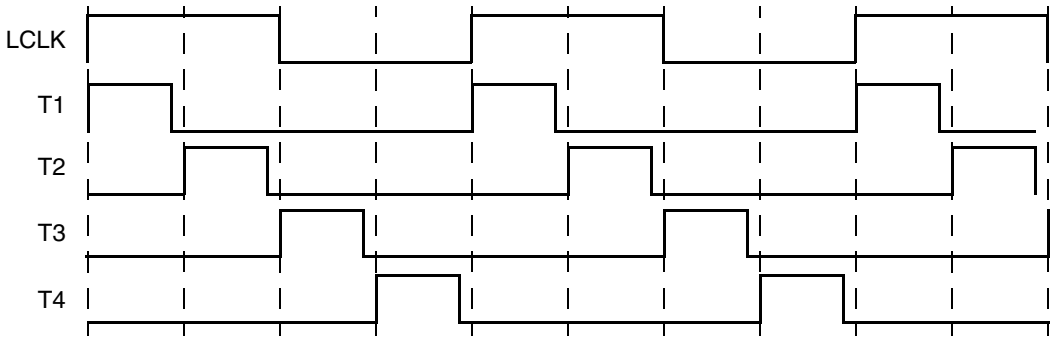


Figure 13-56. UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8

13.4.4.4 RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 13-57. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCS}n}$ is for the bank that matches the current address. The selected $\overline{\text{LBS}}$ is for the byte lanes read or written by the access.

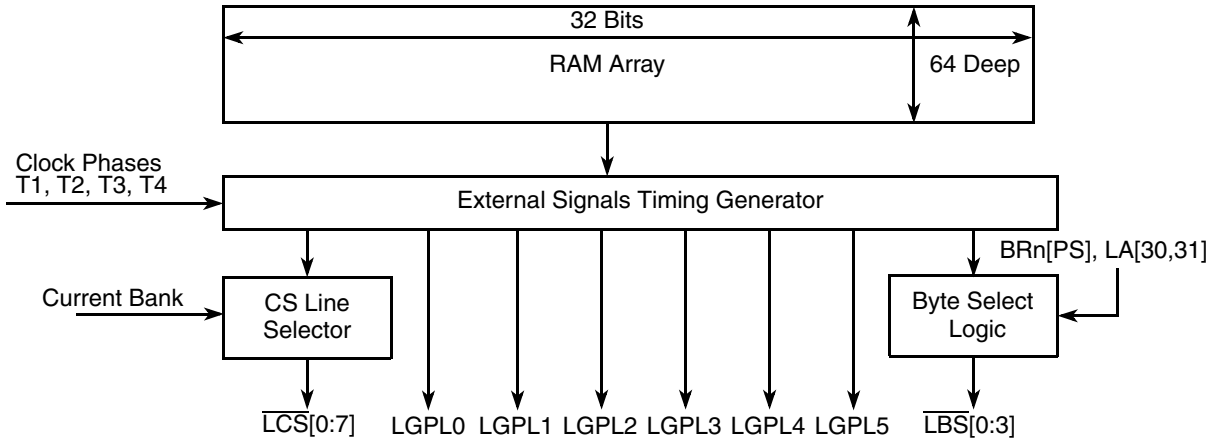


Figure 13-57. RAM Array and Signal Generation

13.4.4.4.1 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 13-58 shows the RAM word fields. When LCRR[CLKDIV] = 4 or 8, the CST*n* and BST*n* bits determine the state of UPM signals $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$[0:3] at each quarter phase of the bus clock. When LCRR[CLKDIV] = 2, CST2 and CST4 are ignored and the external has the values defined by CST1 and CST3 but extended to half the clock cycle in duration. The same interpretation occurs for the BST*n* bits when LCRR[CLKDIV] = 2.

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CST1	CST2	CST3	CST4	BST1	BST2	BST3	BST4	G0L		G0H		G1T1	G1T3	G2T1	G2T3
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	G3T1	G3T3	G4T1/DLT3	G4T3/WAEN	G5T1	G5T3	REDO		LOOP	EXEN	AMX		NA	UTA	TODT	LAST
Reset	All zeros															

Figure 13-58. RAM Word Field Descriptions

Table 13-30 describes RAM word fields.

Table 13-30. RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock quarter phase 1 if LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock half phase 1 if LCRR[CLKDIV] = 2.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ 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 $\overline{\text{LCS}n}$ during bus clock quarter phase 3 if LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock half phase 2 if LCRR[CLKDIV] = 2.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ 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 $\overline{\text{LBS}}$ during bus clock quarter phase 1 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 1 (LCRR[CLKDIV] = 2), in conjunction with BR*n*[PS] and the state of LA[30:31].
5	BST2	Byte select timing 2:. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 2 (LCRR[CLKDIV] = 4 or 8), in conjunction with BR*n*[PS] and the state of LA[30:31]. Ignored when LCRR[CLKDIV] = 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 3 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 2 (LCRR[CLKDIV] = 2), in conjunction with BR*n*[PS] and the state of LA[30:31].
7	BST4	Byte select timing 4. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 4 (LCRR[CLKDIV] = 4 or 8), in conjunction with BR*n*[PS] and the state of 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*x*MR[G0CL] 01 Reserved 10 0 11 1
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*x*MR[G0CL] 01 Reserved 10 0 11 1
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).

Table 13-30. RAM Word Field Descriptions (continued)

Bits	Name	Description
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).
18	G4T1/DLT3	General-purpose line 4 timing 1/delay time 3. The function of this bit is determined by M*x*MR[GPL4]. If M*x*MR[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 M*x*MR[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 M*x*MR[GPL4]. If M*x*MR[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 M*x*MR[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 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 M*x*MR. 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.
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 $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ to prevent data corruption. If EXEN = 0, exceptions are ignored by UPM (but not by 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 13-30. RAM Word Field Descriptions (continued)

Bits	Name	Description
26–27	AMX	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. 00 LAD[0:31] is the non-multiplexed address. For example, column address. 01 Reserved 10 LAD[0:31] is the address multiplexed according to M*x*MR[AM]. For example, row address. 11 LAD[0:31] is the contents of MAR. Used, for example, to initialize a mode. Note that Source ID debug mode is only supported for the AMX = 00 setting.
28	NA	Next burst address. Determines when the address is incremented during a burst access. 0 The address increment function is disabled. 1 The address is incremented in the next cycle. In conjunction with the BR*n*[PS], the increment value of the state of LA[27:31] is 1, 2 or 4 for port sizes of 8-bits, 16-bits and 32-bits, respectively.
29	UTA	UPM transfer acknowledge. Indicates assertion of transfer acknowledge in the current cycle. 0 Transfer acknowledge is not asserted in the current cycle. 1 Transfer acknowledge is asserted in the current cycle.
30	TODT	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 $\overline{\text{RAS}}$ 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 M*x*MR[DS*n*]. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored. 0 The disable timer is turned off. 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.
31	LAST	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 M*x*MR[DS*n*]. 0 The UPM continues executing RAM words. 1 Indicates the last RAM word in the program. The service to the UPM request is done after this cycle concludes.

13.4.4.4.2 Chip-Select Signal Timing (CST*n*)

If BR*n*[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{LCS}n}$ for that bank with timing as specified in the UPM RAM word CST*n* fields. The selected UPM affects only the assertion and negation of the appropriate $\overline{\text{LCS}n}$ signal. The state of the selected $\overline{\text{LCS}n}$ signal of the corresponding bank depends on the value of each CST*n* bit. Figure 13-59 shows how UPMs control $\overline{\text{LCS}n}$ signals.

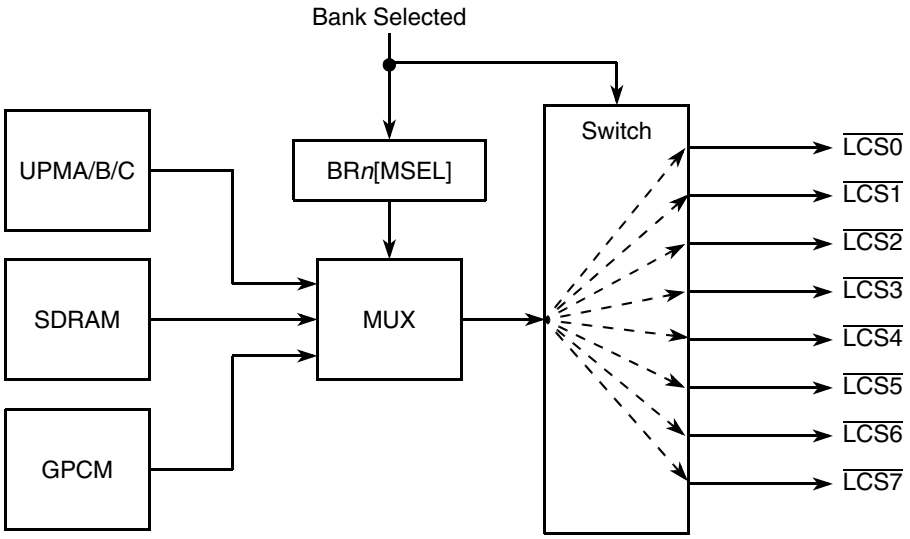


Figure 13-59. $\overline{\text{LCS}}n$ Signal Selection

13.4.4.4.3 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 $\overline{\text{LBS}}$[0:3] signal. The timing of all four byte-select signals is specified in the RAM word. However, $\overline{\text{LBS}}$[0:3] are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

Figure 13-60 shows how UPMs control $\overline{\text{LBS}}$[0:3].

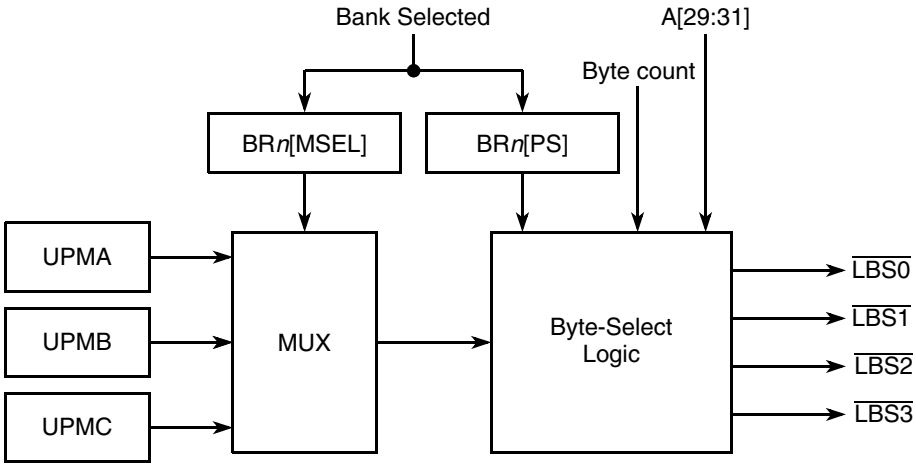


Figure 13-60. $\overline{\text{LBS}}$ Signal Selection

The uppermost byte select ($\overline{\text{LBS0}}$), when asserted, indicates that LAD[0:7] contains valid data during a cycle. Likewise, $\overline{\text{LBS1}}$ indicates that LAD[8:15] contains valid data, $\overline{\text{LBS2}}$ indicates that LAD[16:23] contains valid data, and $\overline{\text{LBS3}}$ 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, $\overline{\text{LBS}}$[0:3] signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

13.4.4.4.4 General-Purpose Signals (G*n*T*n*, 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 LGPL*n* lines.

GPL0 can be controlled by an address line specified in M*x*MR[G0CL]. To use this feature, G0H and G0L 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.

13.4.4.4.5 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 13-31. 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 if LAST and LOOP must not be set together.

Table 13-31. M*x*MR 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

13.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long UPM 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.

13.4.4.4.7 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*x*MR[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 lsbs 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 13-32 shows how M*x*MR[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 13-32. 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 the AMX field from one RAM word to the next RAM word executed results in an address phase on the LAD[0:31] bus with the assertion of LALE for the number of cycles set for LALE in the OR*n* and LCRR registers. The LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

13.4.4.4.8 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*x*MR[GPL*n*4DIS], determines when the data input is sampled by the LBC as follows:

- If M*x*MR[GPL*n*4DIS] = 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*n*4DIS = 0 (G4T4/DLT3 functions as G4T4), or if GPL*n*4DIS = 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 13-61 shows how data sampling is controlled by the UPM.

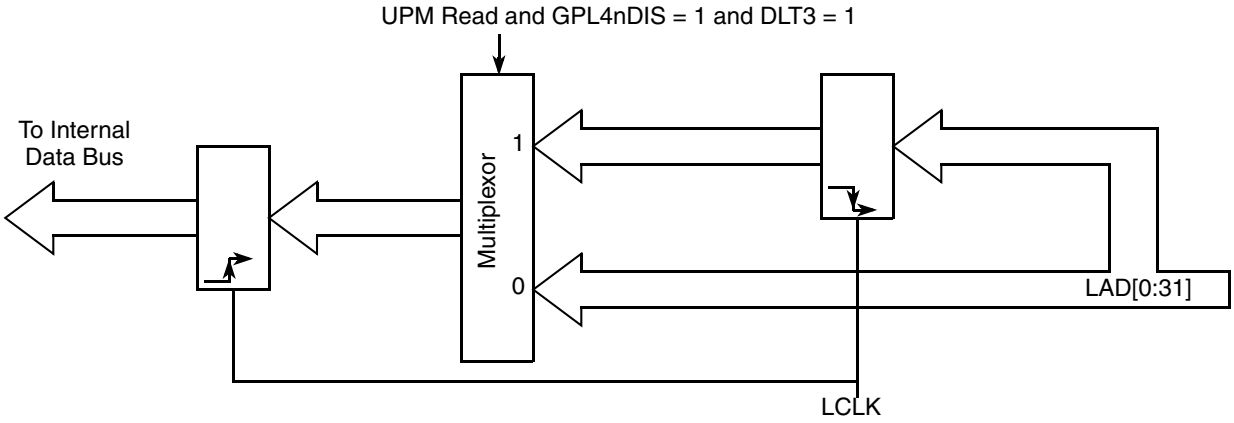


Figure 13-61. UPM Read Access Data Sampling

13.4.4.4.9 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 the UPM signals are negated unconditionally (driven to logic 1), 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.

13.4.4.4.10 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. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 13-62 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 $\overline{\text{LCS}n}$ and LGPL1 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.

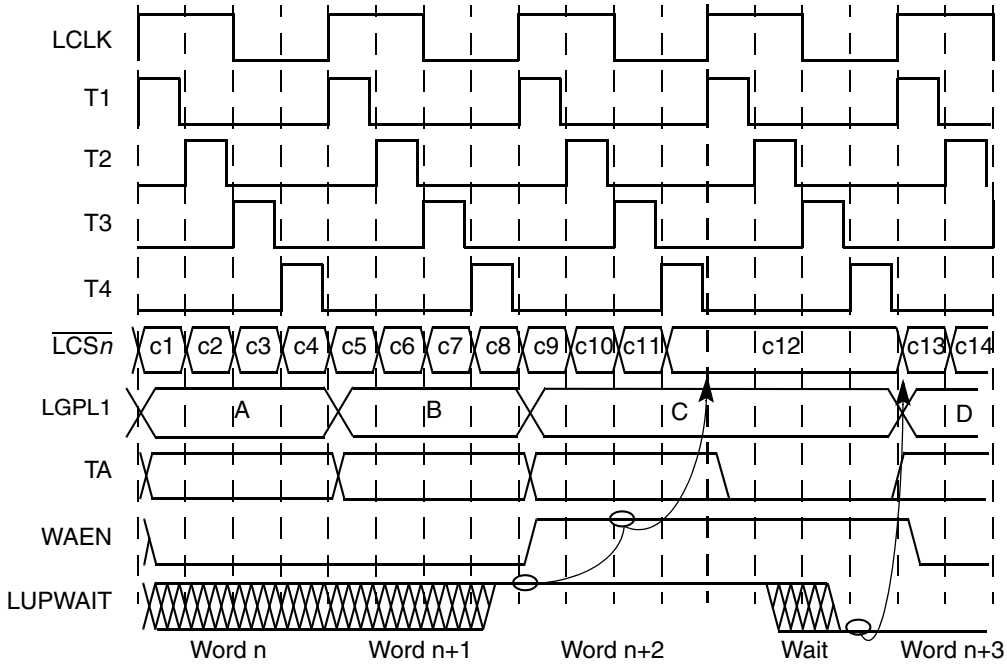


Figure 13-62. Effect of LUPWAIT Signal

13.4.4.5 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.

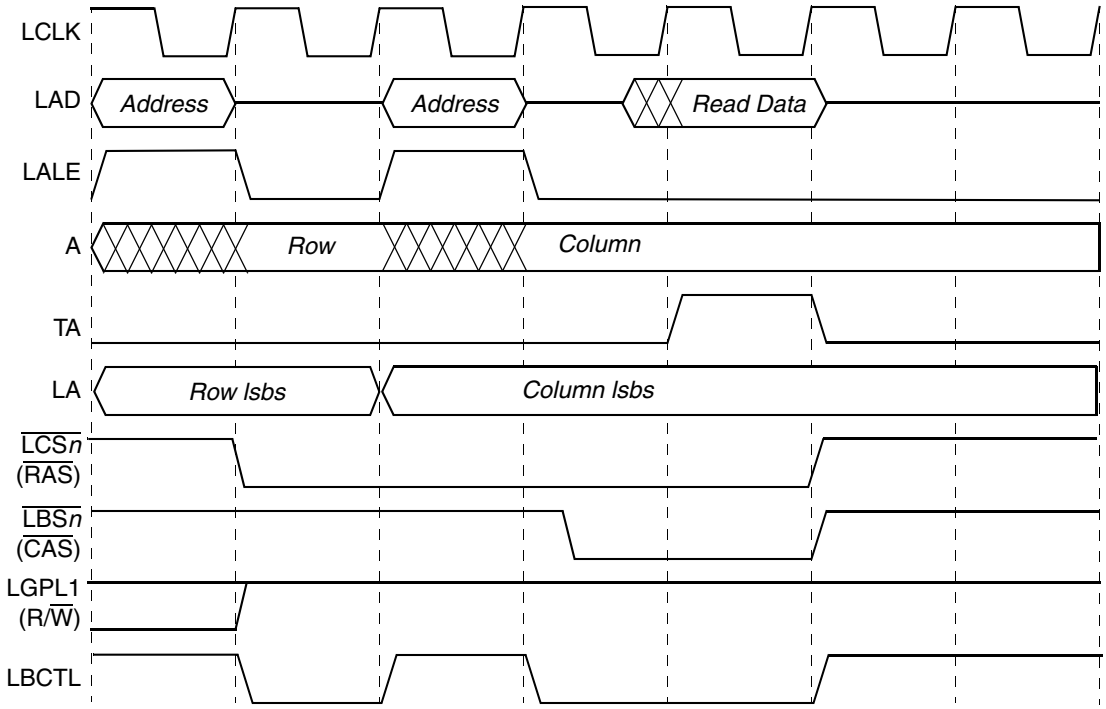
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately 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 UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to affect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

13.4.4.6 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 OR*n*[TRLX] and OR*n*[EHTR]. The next accesses after a read access to the slow memory device is delayed by the number of clock cycles specified in the OR*n* register in addition to any existing bus turn around cycle.

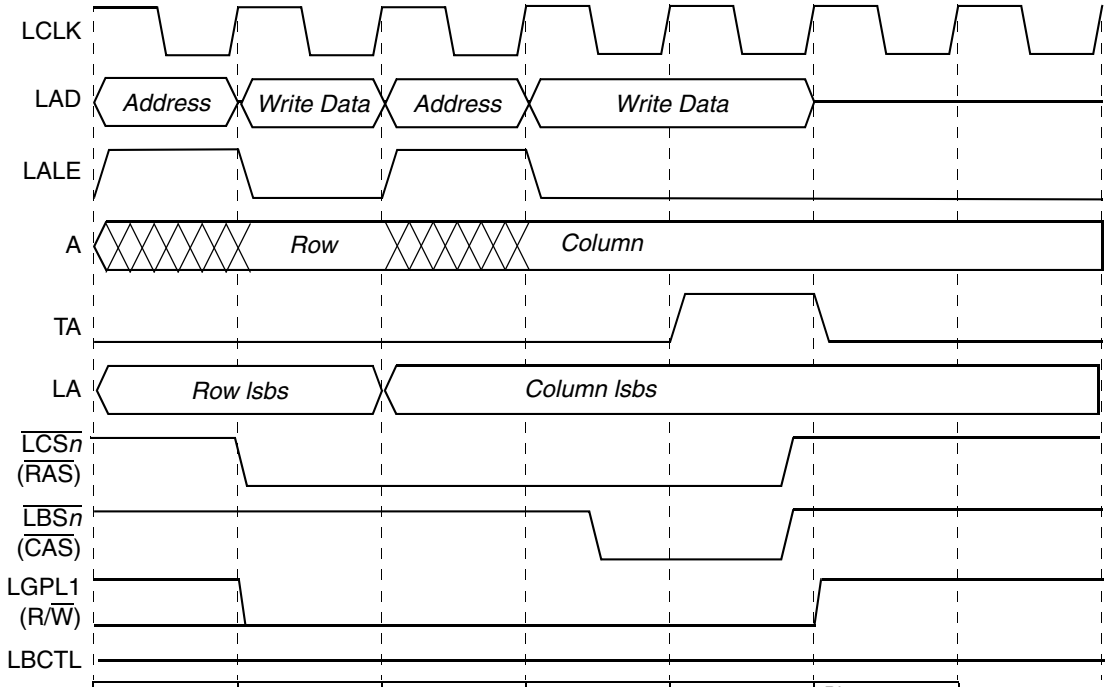
13.4.4.7 Memory System Interface Example Using UPM

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 shows timing diagrams for various UPM configurations, using fast-page mode DRAM as an example, with LCRR[CLKDIV] = 4 or 8. These illustrative 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 LGPL*n* signal may be used for this purpose.



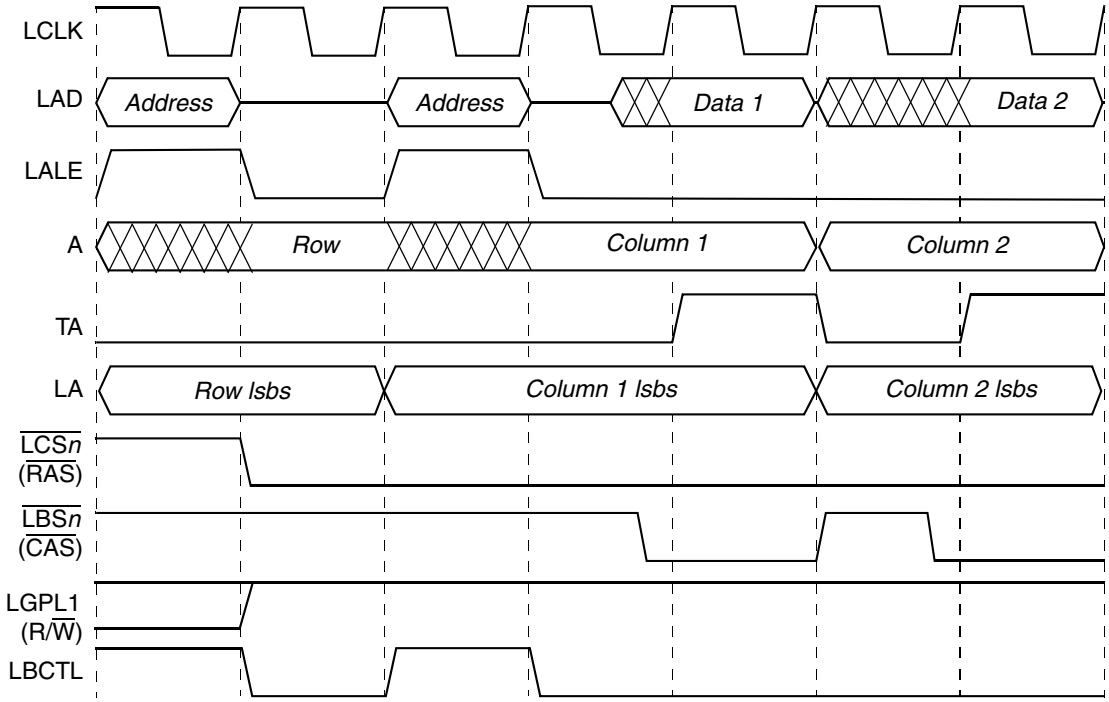
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	0	Bit 3
bst1	1		1	0	Bit 4
bst2	1		0	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	0	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	1		1	1	Bit 12
g1t3	1		1	1	Bit 13
g2t1					Bit 14
g2t3					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 13-63. Single-Beat Read Access to FPM DRAM



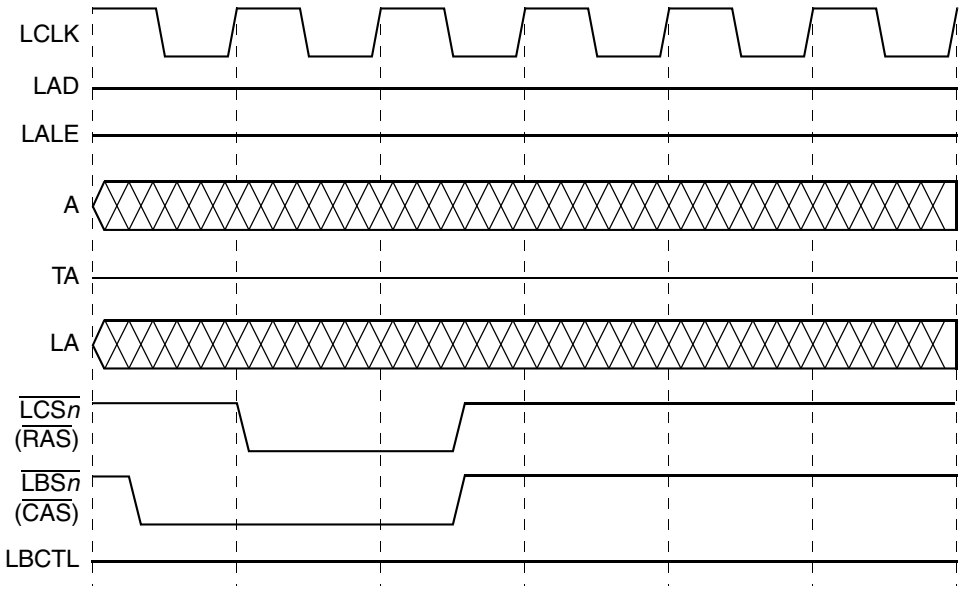
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 0 0	0 0 0 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	0 0		0 0	0 0	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit 16 Bit 17
g4t1 g4t3					Bit 18 Bit 19
g5t1 g5t3					Bit 20 Bit 21
redo[0] redo[1]					Bit 22 Bit 23
loop exen	0 0		0 0	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0		0 0 0 0 0 0	0 0 0 1 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	WSS	**WSS+1**	**WSS+1**	**WSS+2**	

Figure 13-64. Single-Beat Write Access to FPM DRAM



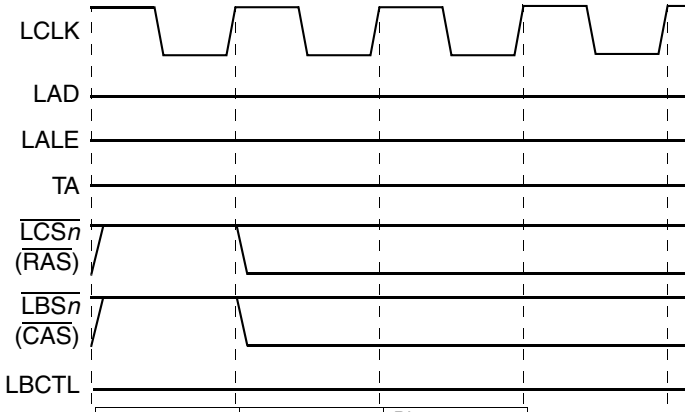
cst1	0	LALE pause (due to change in AMX)	0	0	1	Bit 0
cst2	0		0	0	1	Bit 1
cst3	0		0	0	1	Bit 2
cst4	0		0	0	1	Bit 3
bst1	1		1	0	1	Bit 4
bst2	1		1	0	1	Bit 5
bst3	1		1	0	1	Bit 6
bst4	1		0	0	1	Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1		1	1	1	Bit 12
g1t3	1		1	1	1	Bit 13
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 13-65. 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 13-66. 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
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	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 13-67. Exception Cycle

13.5 Initialization/Application Information

13.5.1 Interfacing to Peripherals

13.5.1.1 Multiplexed Address/Data Bus and Non-Multiplexed 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 the LALE signal 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 non-multiplexed. For 8-bit devices, LA[30:31] should be used and for 16-bit devices, LA[30] 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 five least significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

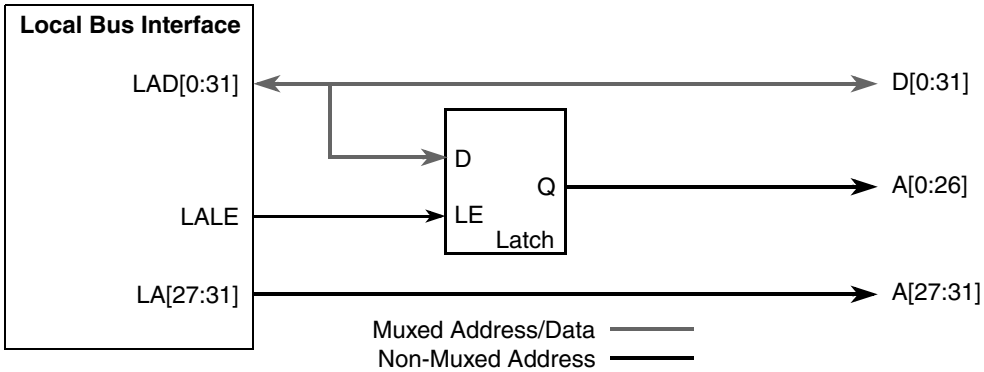


Figure 13-68. Multiplexed Address/Data Bus

13.5.1.2 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 13-69.

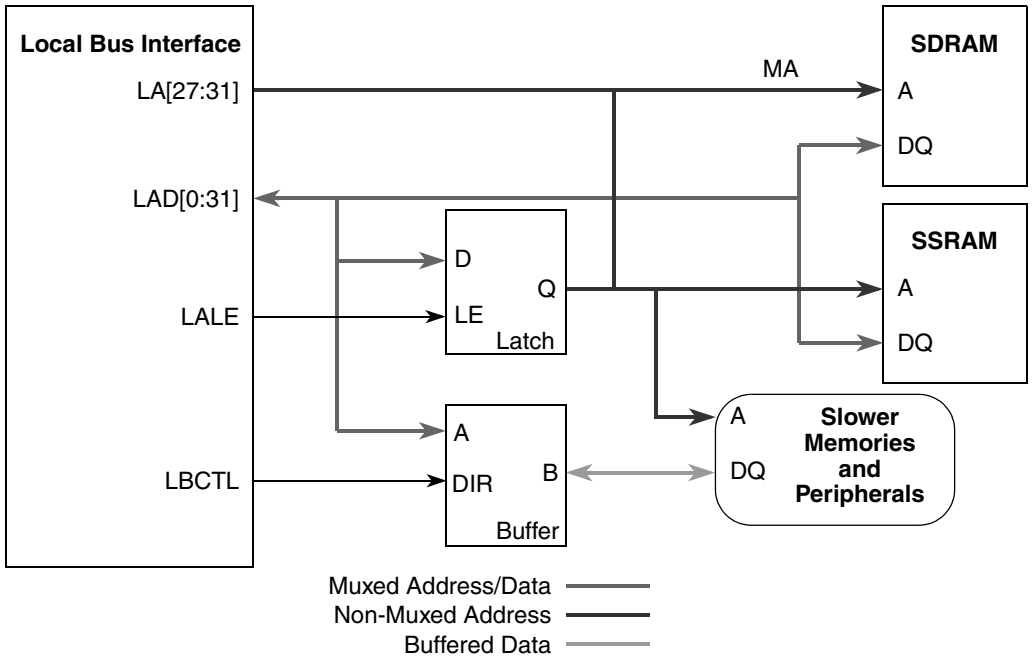


Figure 13-69. Local Bus Peripheral Hierarchy

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.

13.5.1.3 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 13-70 shows an example of such a hierarchy. This section is only a guideline and the board designer must simulate the electric characteristics of his scenario to determine the maximum operating frequency.

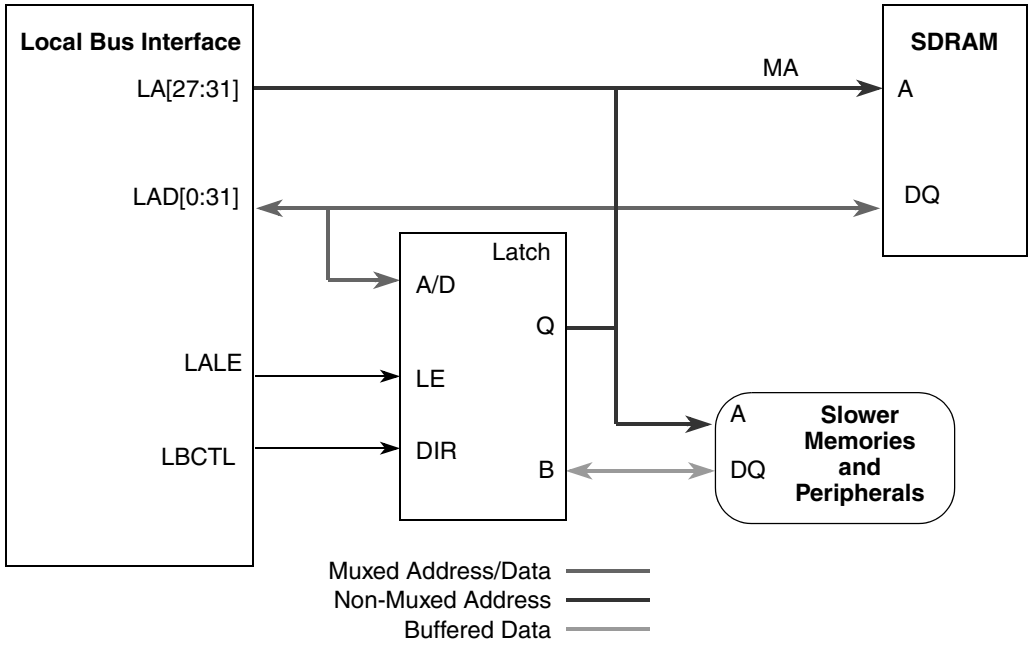


Figure 13-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

13.5.1.4 GPCM Timings

In case a system contains a memory hierarchy with high speed synchronous memories (SDRAM, synchronous SRAM) and lower speed asynchronous memories (for example, 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.

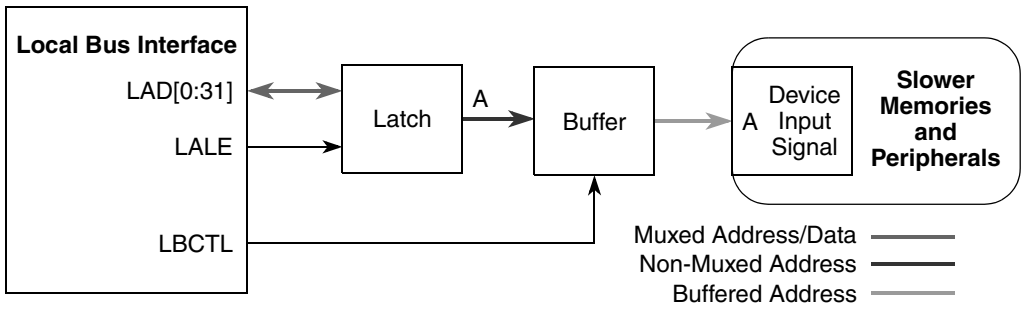


Figure 13-71. GPCM Address Timings

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 2 propagation delays are in the order of 3–6 ns, so for a 166-MHz bus frequency, $\overline{\text{LCS}}$ should arrive on the order of 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time has to be considered.

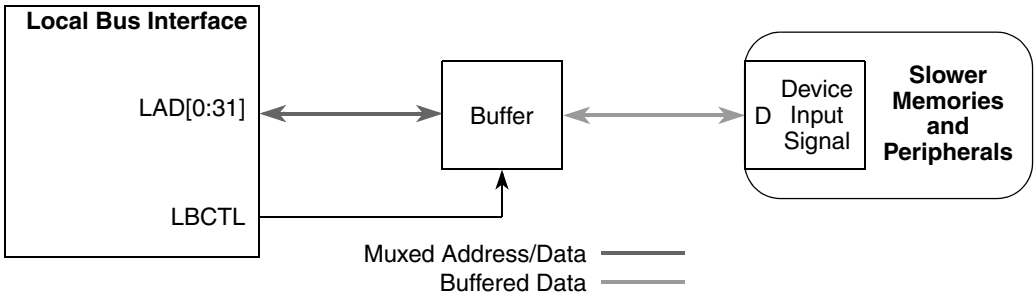


Figure 13-72. GPCM Data Timings

13.5.2 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

13.5.2.1 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.

13.5.2.2 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}(transceiver)] is larger than t_{dis} (LB) to avoid bus contention.

13.5.2.3 Read-Modify-Write Cycle for Parity Protected Memory Banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle will have a new address phase in any case, this basically is the same case as an address phase after a previous read.

13.5.2.4 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.

13.5.3 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 13-73 shows the device connections on the data bus.

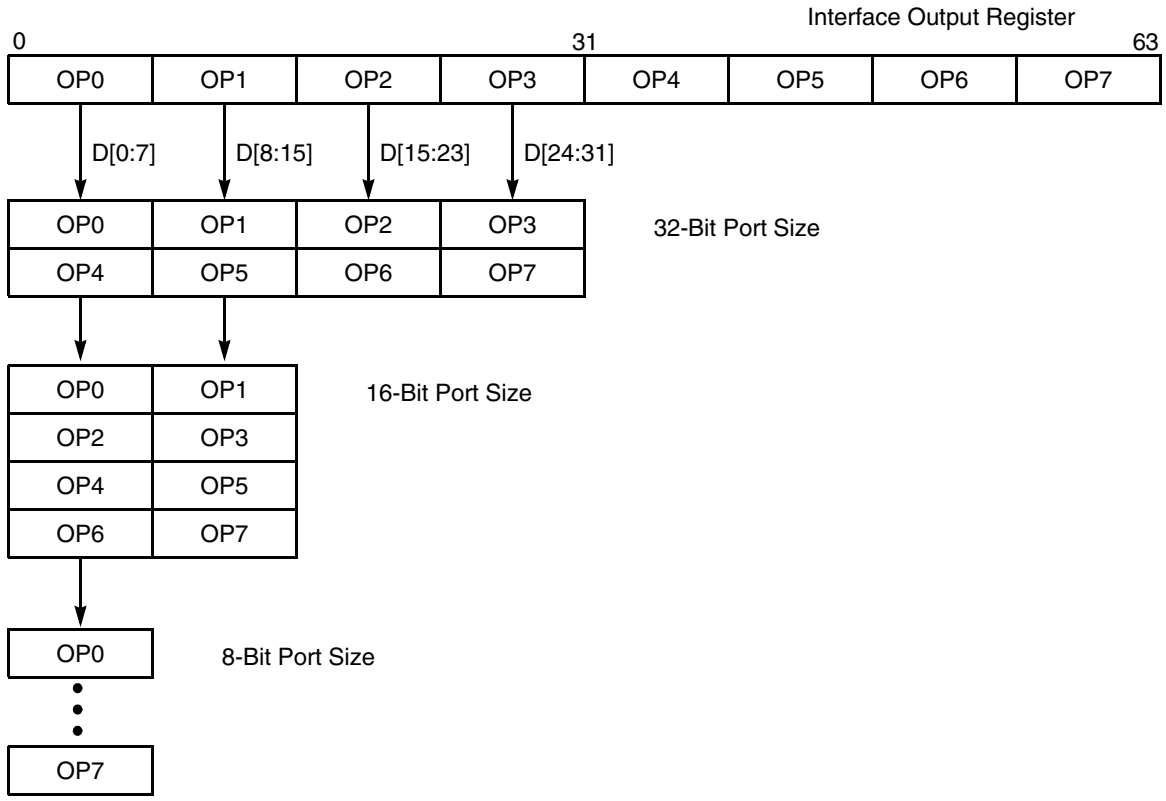


Figure 13-73. Interface to Different Port-Size Devices

Table 13-33 lists the bytes required on the data bus for read cycles.

Table 13-33. 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 13-33. 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.

13.5.4 Interfacing to SDRAM

The following sections provide application information on interfacing to SDRAM.

13.5.4.1 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 ($\overline{\text{LCS}}n$) 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 $\overline{\text{LCS}}$[1:7] can be programmed to support SDRAM, assuming $\overline{\text{LCS0}}$ 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 exactly 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:

$$\text{Refresh Period} = \frac{\text{LSRTx (MPTPR[PTP])}}{\text{System Frequency}}$$

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 13-51 in Section 13.4.3.11.1, "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.

13.5.4.2 Maximum Amount of SDRAM Supported

Table 13-34 summarizes information based on typical SDRAM data sheets.

Table 13-34. Typical SDRAM Devices

SDRAM Device	64-Mbit				128-Mbit				256-Mbit				512-Mbit			
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	—
Column	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	—

The data port size is programmable, but the following examples use all 32 bits of the local bus. The 32-bit port size requires 4 SDRAM devices (with 8-bit I/O ports) connected in parallel to a single chip select. If 128-Mbit devices are used, 1 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 Mbytes. If 256-Mbit SDRAM devices are used, the total available memory is 512 Mbytes. 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.

13.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

13.5.4.3.1 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.

13.5.4.3.2 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.

13.5.4.3.3 128-Mbyte SDRAM

Figure 13-74 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.

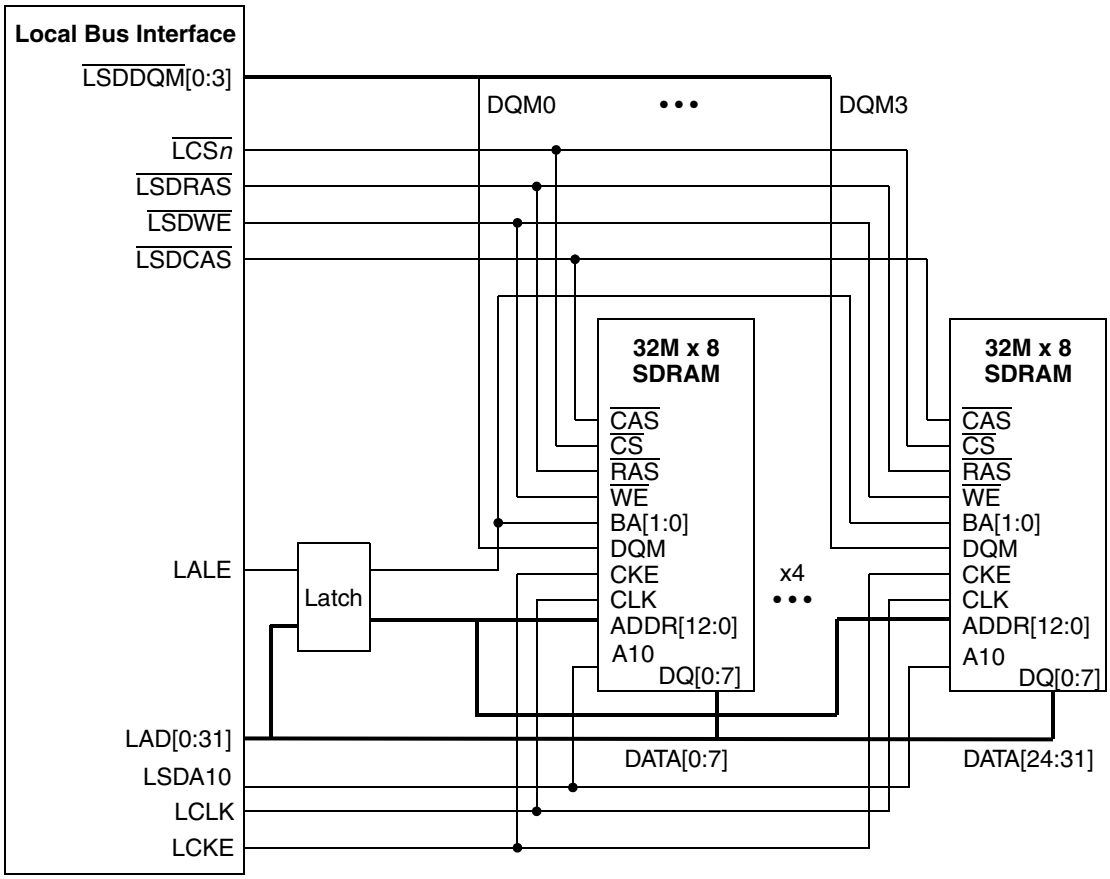


Figure 13-74. 128-Mbyte SDRAM Diagram

Table 13-35 shows details about LAD*n* signal connections for the example in Figure 13-74.

Table 13-35. 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 13-35. 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 × 8 × 32 Mbits.
- Each device has 4 internal banks, 13 row address lines, and 10 column address lines.

The logical address is partitioned as shown in Table 13-36.

Table 13-36. 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 13-37.

Table 13-37. 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 13-38 shows the address port configuration during a READ/WRITE command.

Table 13-38. 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 13-39 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 13-39. Register Settings for 128-Mbyte SDRAMs

Register	Field	Value
BR*n*	BA PS MS V	Base address 11 = 32-bit port size 011 = SDRAM-local bus 1
OR*n*	AM COLS ROWS	11_1111_1000_0000_0000_0 011 100
LSDMR	RFEN OP BSMA RFRC PRETOACT ACTTOROW BL WRC BUFCMD CL	1 000 011 From device data sheet From device data sheet From device data sheet 0 From device data sheet 0 From device data sheet

13.5.4.3.4 256-Mbyte SDRAM

This example uses the same Micron SDRAM as in the previous example, but doubles the number of devices connected and therefore uses two chip selects.

13.5.4.3.5 512-Mbyte SDRAM

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is 8 × 4 × 64 Mbit × 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 13-40.

Table 13-40. 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 13-41.

Table 13-41. 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 13-42 shows the address port settings during a READ/WRITE command.

Table 13-42. 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 13-43 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

Table 13-43. Register Settings for 512-Mbyte SDRAMs

Register	Field	Value
BR*n*	BA PS MS V	Base address 11 = 32-bit port size 011 = SDRAM-local bus 1
OR*n*	AM BPD COLS ROWS	11_1110_0000_0000_0000_0 01 100 100
PSDMR	RFEN OP BSMA RFRC PRETOACT ACTTOROW BL WRC BUFCMD CL	1 000 011 From device data sheet From device data sheet From device data sheet 0 From device data sheet 0 From device data sheet

13.5.4.3.6 Power-Down Mode

SDRAMs offer a power-down mode during which the device is not refreshed, and therefore, data is not maintained. This mode is invoked by driving CKE low, while all internal banks are idle; note that they must be precharged first. Figure 13-75 shows the timing. Note that the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command.

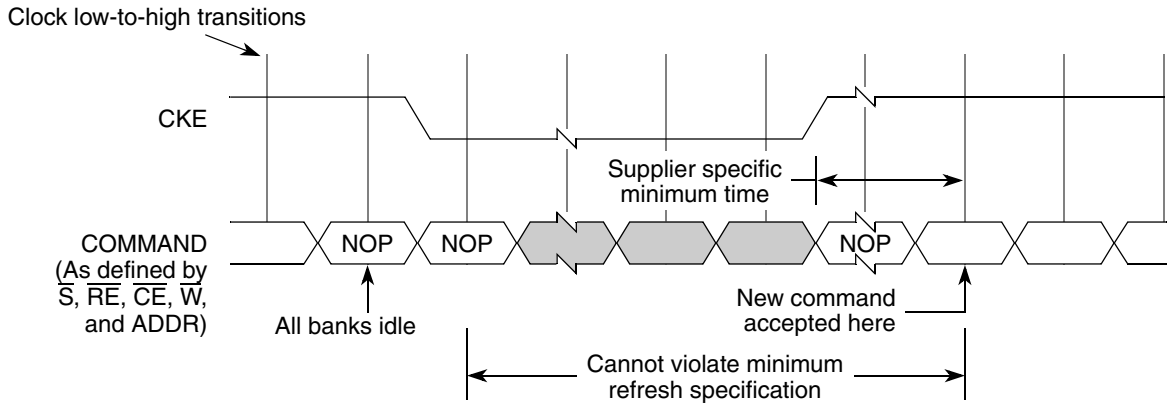


Figure 13-75. SDRAM Power-Down Timing

CKE remains low, as long as the device is powered down. After CKE transitions to high, the SDRAM exits the power-down mode.

13.5.4.3.7 Self-Refresh

In order to be able to stop activity on the local bus (for power save or debug), while the content of the SDRAM is maintained, the self-refresh mode is supported. This mode is invoked by issuing a self-refresh command to the SDRAM. The LBC applies the same timing as for the auto refresh, but also pulls the SDRAM CKE (LCKE) signal low in the same cycle. This can only be done with all banks being idle; the SDRAM machine must precharge them ahead of this. As long as CKE stays low, the device refreshes itself and does not need to see any refreshes from the local bus. To exit self refresh, CKE simply has to be pulled high. Note that after returning from self-refresh mode the SDRAM needs a supplier-specific time before it can accept new commands and the auto-refresh mechanism has to be started again. Figure 13-76 shows this timing. The SDRAM controller always uses 200 local bus clocks, which should satisfy any SDRAM requirements. As in the case of the power-down mode, the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command.

1. Refer to Section 13.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands,” for SDRAM interface commands and information on the self-refresh command.

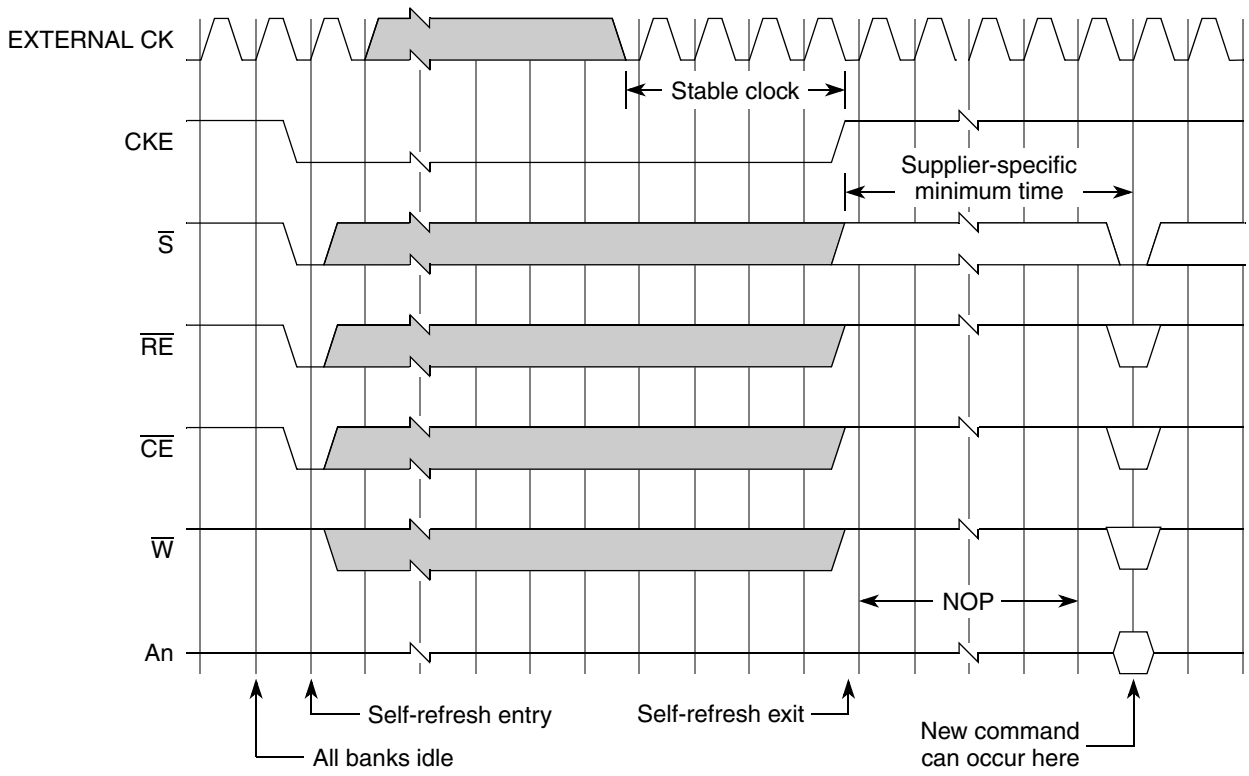


Figure 13-76. SDRAM Self-Refresh Mode Timing

13.5.4.3.8 SDRAM Timing

To allow for very high speeds on the memory bus, the capacitive loading on the local bus must be taken into consideration as shown in Table 13-44.

Table 13-44. SDRAM Capacitance

Signal	Min	Max	Unit
CLK	2.0	4.0	pf
$\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, $\overline{CS}$, CKE, DQM	2.0	5.0	pf
Address	2.0	5.0	pf
DQ_0–DQ_{31}	3.5	6.5	pf

Note: Capacitance values compiled from worst case numbers from various data sheets from Samsung and Micron

To implement a system using the hierarchy described earlier for 2 synchronous memory banks, 1 address latch, and 1 buffer loading the multiplexed address/data bus sees a loading of 4 loads of about 6.5 pF maximum. For a nominal load, 30 pF can be used.

Table 13-45. SDRAM AC Characteristics

Parameter		Device Speed				Unit
		166 MHz		133 MHz		
		Min	Max	Min	Max	
CLK cycle time	CAS latency = 3	6	1000	7.5	1000	ns
	CAS latency = 2	—		7.5		
CLK to valid output delay	CAS latency = 3	—	5	—	5.4	ns
	CAS latency = 2	—	—	—	5.4	
Output data hold time	CAS latency = 3	2.5	—	3	—	ns
	CAS latency = 2	—	—	3	—	
Input setup time		1.5	—	2	—	ns
Input hold time		1	—	1	—	ns
CLK to output in Hi-Z	CAS latency = 3	—	5	5.4	—	ns
	CAS latency = 2	—	—	5.4	—	

Note: AC characteristics compiled from worst-case numbers from various data sheets from Samsung and Micron.

Setup and Hold Timing Calculations:

Address TOF (time of flight): board layout delay

Data TOF (time of flight): board layout delay

Clock skew (time of flight): clock skew between the LBC and the clock at the memory device. The local bus DLL feedback mechanism must be used to control this skew to optimize the timing margins, as described in the rest of this subsection.

Address setup margin = cycle time – local bus address CTQ – SDRAM address input setup time – address TOF + clock skew

Address hold margin = local bus address output hold time + address TOF – SDRAM address input hold time – clock skew

Data write to SDRAM setup margin = cycle time – local bus data CTQ – SDRAM data input setup time – data TOF + clock skew

Data write to SDRAM hold margin = local bus data output hold time + data TOF – SDRAM data input hold time – clock skew

Data read from SDRAM setup margin = cycle time – SDRAM data CTQ – local bus data input setup time – data TOF – clock skew

Data read from SDRAM hold margin = SDRAM data output hold time + data TOF – local bus data input hold time + clock skew

To improve the timing margins a DLL is used to generate external clocks, which minimize the skew between the local bus and the memory clock. Figure 13-77 shows relative timings for the local bus clock DLL.

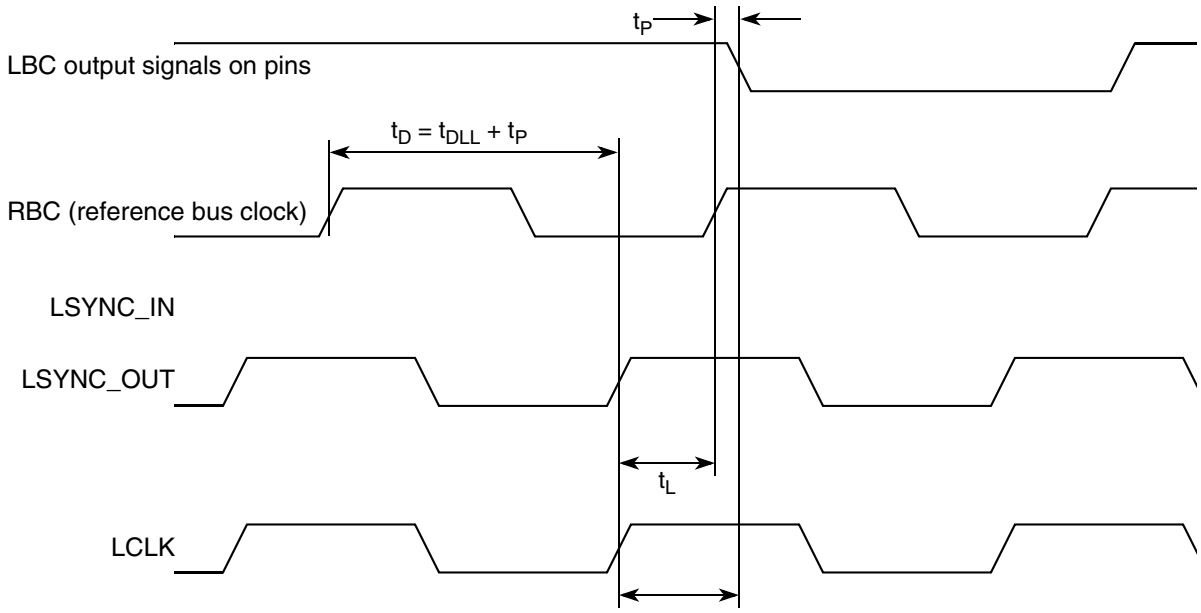


Figure 13-77. Local Bus DLL Operation

13.5.4.4 Parity Support for SDRAM

Contrary to older DRAM technologies, SDRAM devices typically are organized either x4, x8, x16, or x32. There are no mainstream devices that include parity support. To allow for error protection on the local bus an additional SDRAM for the 4 parity bits must be used. Since 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 13-78 shows a connection diagram.

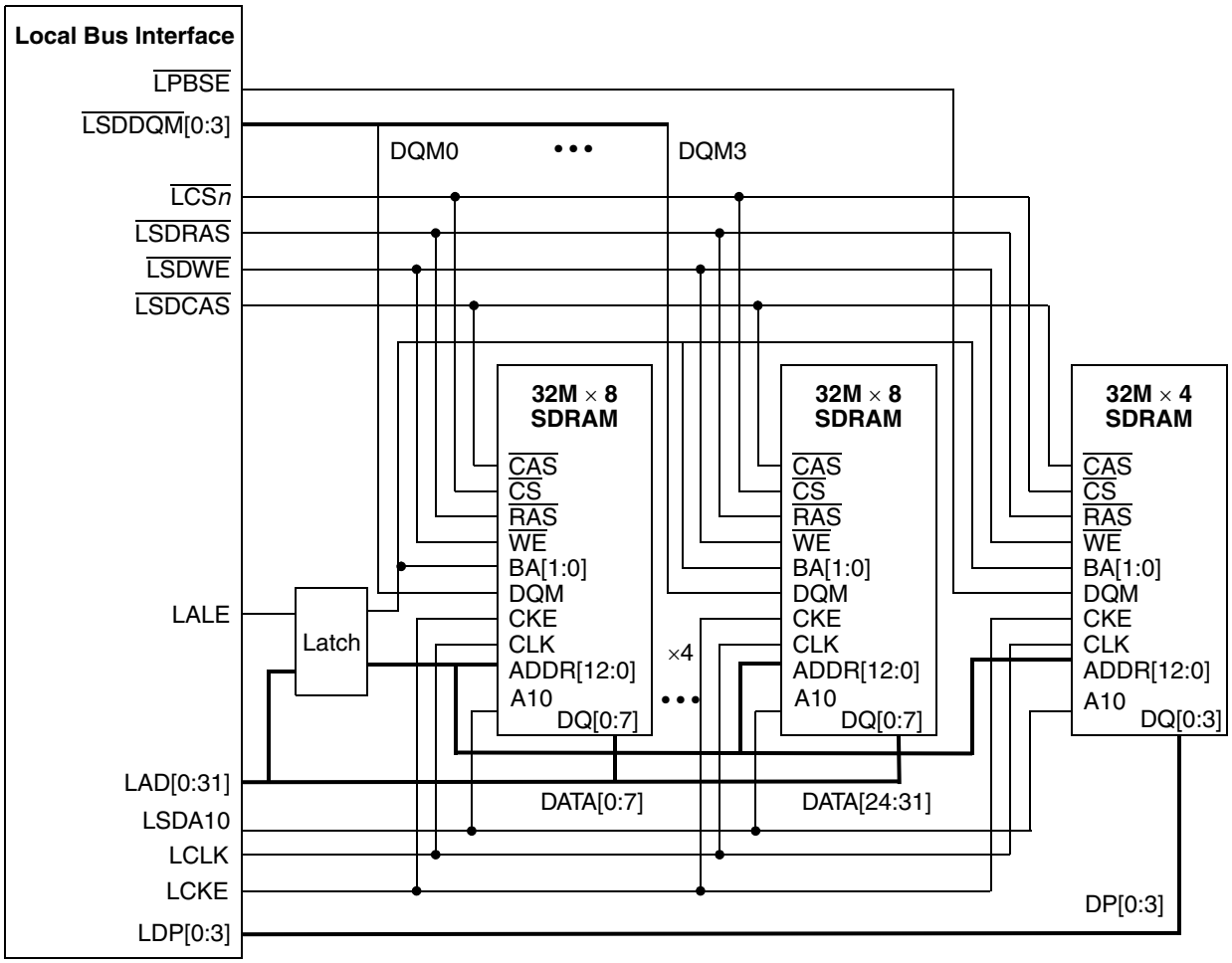


Figure 13-78. Parity Support for SDRAM

13.5.5 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 13-79 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 will mostly be used by performance-critical applications, we assume here that, typically, the maximum width of the local bus of 32 bits will be used.

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.

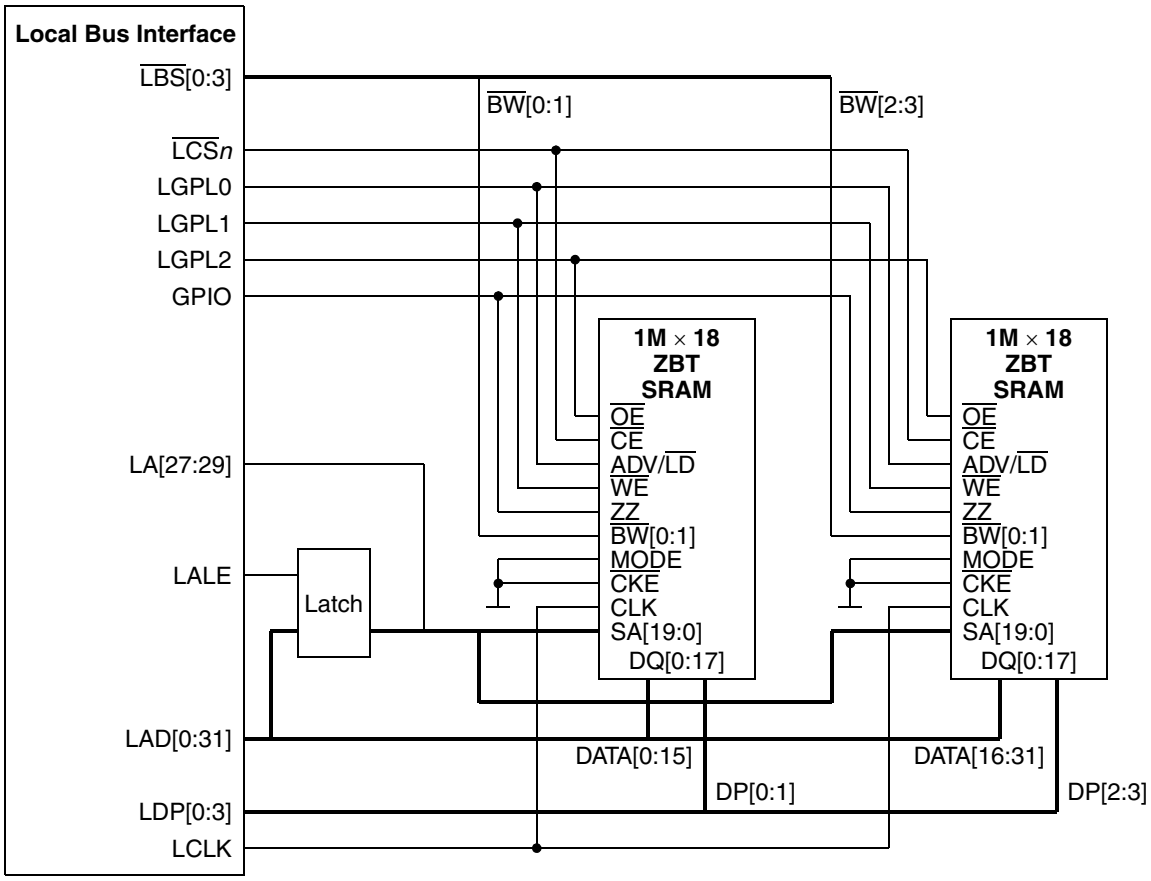


Figure 13-79. Interface to ZBT SRAM

Because we use linear burst on the SRAM, the device will itself burst 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 has to 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 $\overline{\text{WE}}$). The UPM controller basically has to wait for the end of the SRAM burst to avoid bus contention with further bus activities.

ZBT SRAMs have a power down mode, which is invoked by the ZZ signal. Connecting a GPIO signal to ZZ allows use of that power down mode; however, accesses to the SRAM while in power down mode do not create valid results. This should be taken care of by the system software.

Another observation is that SRAMs are available with natural parity. In the example, we use a ×18 SRAM, which holds two data bytes and two parity bits. While for the support of parity on SDRAM banks the local

bus has to use read-modify-write cycles and compromise performance, SRAM banks can be used with natural parity and do not compromise performance for parity support.

13.5.6 Interfacing to DSP Host Ports

In many applications, an integrated communications processor aggregates traffic for DSPs and distributes that traffic to the DSPs. The local bus allows connection to a variety of different DSP host ports and this section gives some information on how to interface to some example DSPs.

13.5.6.1 Interfacing to MSC8101 HDI16

This section describes how to interface to the HDI16 peripheral interface of the MSC8101. After initial set-up of the interface, the host and HDI16 device can communicate either by a read and write transaction from the core or, if the setup on the DSP and the host are implemented appropriately, by DMA transfers of the host DMA controller, which can be triggered automatically by signals generated by the HDI16 peripheral.

13.5.6.1.1 HDI16 Peripherals

The host interface (HDI16) is a 16-bit-wide, full-duplex, double-buffered parallel port that can directly connect to the data bus of a host processor. It supports a variety of buses and gluelessly connects with a number of industry-standard microcomputers, microprocessors, and DSPs. The HDI16 also supports the 8-bit host data bus, which makes it fully compatible with the DSP56300 HI08 (as viewed by the host side, not from the DSP side).

The host bus can operate asynchronously to the SC140 core clock, and the HDI16 registers are divided into two banks. The host register bank is accessible to the external host, and the core register bank is accessible to the SC140 core.

The MSC8101HDI16 host port peripheral has two sets of 16-bit-wide registers—one set is only visible internally to the DSP, while the other set is visible only to the external host processor. Figure 13-81 illustrates the relationship between the two sides.

All of the HDI16 peripheral's registers are mapped directly onto the MSC8101 QBus, as defined by the *MSC8101 16-Bit Digital Signal Processor Reference Manual* (MSC8101RM); the transmit and receive FIFOs are mapped onto the DMA data bus such that the DMA controller can access them directly without core intervention. The addressing for each of these registers is defined in Section 13.5.6.1.2, "Physical Interconnections."

The HDI16 host port itself is a 16-bit-wide parallel port with various strobe and multiplexing options.

The most important HDI16 host port facet is that it is specified as an asynchronous interface and so reduces concerns over clock skew between the HDI16 host port and the host device's buses. Furthermore, with all the host port registers being accessed with a single chip select and four address lines, as far as the local bus is concerned, the DSP host port is akin to an asynchronous memory mapped region. So, for the HDI16 port in single strobe mode, the host device asserts a chip select, a single data strobe and a read/write line to select HDI16 read or write bus operations.

The read and write strobes are also used as the data latch control to complete the bus transactions, obviating the need for any handshake termination signal from the DSP. The UPM programmer is responsible for satisfying the AC timings of the HDI16 transactions.

Through appropriate mode selection, the HDI16 peripheral's feature set can be fully supported by the local bus's UPM controlled signals. The UPM defined interface can be used with any of the local bus's eight chip selects to give the 16-bit port size and strobe generation that matches that of the HDI16 host port.

Figure 13-80 shows the internal register diagram of the HDI16.

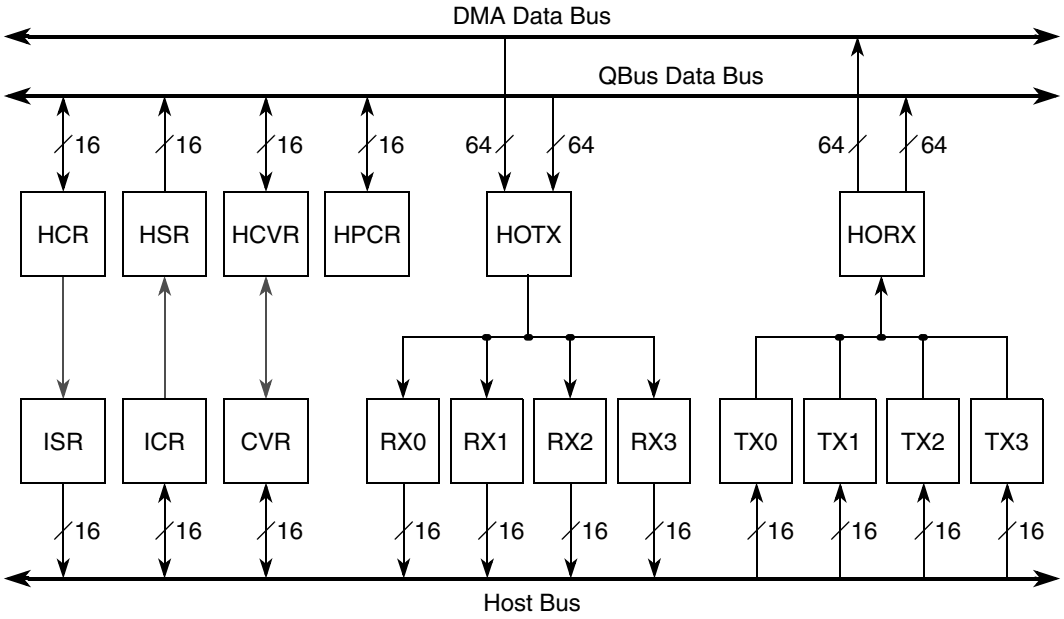


Figure 13-80. MSC8101 HDI16 Peripheral Registers

13.5.6.1.2 Physical Interconnections

The physical interconnections between the UPM controlled local bus and the HDI16 of the MSC8101 HDI16 peripheral are given in Table 13-46 and Figure 13-81.

Table 13-46. Local Bus to MSC8101 HDI16 Connections

MSC8101 Signal(s)	Type	Description	Connect with Local Bus Signal
HD[0:15]	I/O/Z	Host data bus	LAD[0:15]
HA[0]	I	Host address line	Either LA[27] or latched A25
HA[1]	I	Host address line	Either LA[28] or latched A26
HA[2]	I	Host address line	LA[29]
HA[3]	I	Host address line	LA[30]
$\overline{\text{HCS1}}$	I	Host chip select 1	$\overline{\text{LCS}n}$
$\overline{\text{HCS2}}$	I	Host chip select 2	Tie this to V_{DD}
HRDRW	I	Host R/$\overline{\text{W}}$ signal	LGPL1 or inverted LBCTL signal

Table 13-46. Local Bus to MSC8101 HDI16 Connections (continued)

MSC8101 Signal(s)	Type	Description	Connect with Local Bus Signal
$\overline{\text{HDS}}$	I	Host data strobe signal	LGPLy
HRRQ/HACK	O	Receive host request OP	As required in application
HTRQ/HREQ	O	Transmit host request OP	As required in application

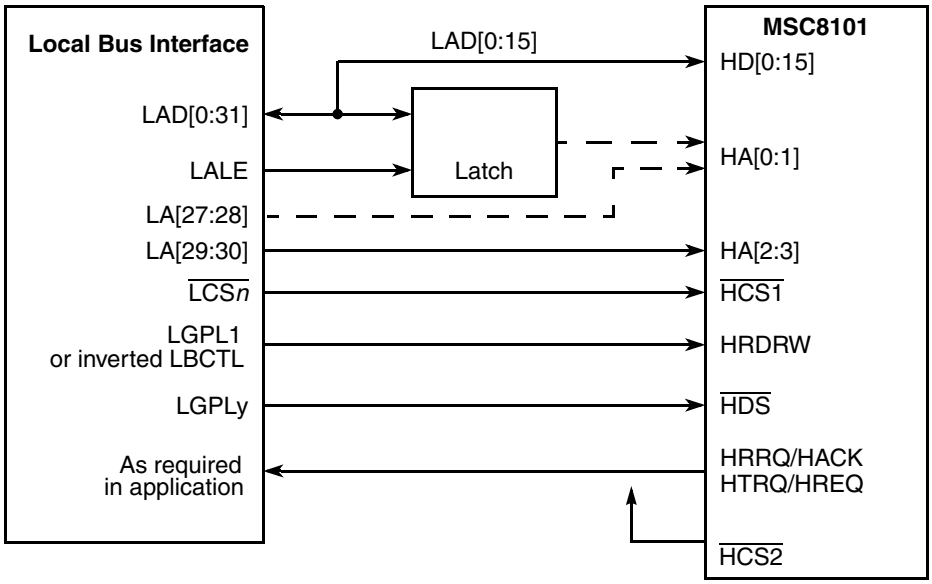


Figure 13-81. Interface to MSC8101 HDI16

The connections are specified as follows:

- One chip select, $\overline{\text{LCS}n}$ (whatever is available in the system), is used to memory map accesses from the host local bus to the HDI16 MSC8101 HDI16 peripheral, and is connected to the HDI16 chip select line (HCS).
- This interface uses two separate general-purpose strobe lines (LGPL1 and LGPLy):
  - LGPL1 is programmed to generate the HDI16 read/write signal (HRDRW), which is typically high for a read access and low for write access. In any case, the host port requires a HR/$\overline{\text{W}}$ signal. This can be generated by using LGPL1, and allows it to adopt the timing virtually without restrictions. Alternatively the designer can invert LBCTL to generate this signal. It is the responsibility of the UPM pattern designer to plan for the additional delay of that inverter in the UPM pattern to satisfy the AC timings at the DSP host port.
  - LGPLy is programmed to generate the HDI16 data strobe ($\overline{\text{HDS}}$), which must be asserted every 16-bit read or write transaction.
- Data lines—The bus data lines ($\overline{\text{LAD}n}$) are directly connected to the HDI16 data lines ($\overline{\text{HD}n}$).
- DMA request/service request signals (HRRQ and HTRQ)—as appropriate in the application

- Address lines—The address lines between the LBC and the HDI16 MSC8101 can either be connected in a straightforward or a specific manner to enable burst transfers across the HDI16. The connections are defined as follows and are described in more detail in the following sub-section:
  - — Either LA[27] or latched value of A25 -> HA0
  - — Either LA[28] or latched value of A26 -> HA1
  - — LA[29] -> HA2
  - — LA[30] -> HA3
- Ground lines—In order to provide the best ground plane, it is highly advised that all grounds are common and connected together.

13.5.6.1.3 Supporting Burst Transfers

As mentioned previously, to facilitate burst transfers the host's local bus address lines can be connected in a very specific way. First, the local bus A31 signal is not required, as the HDI16 registers are 16-bit word addressed. Secondly, local bus LA[27] and LA[28] signals can be eliminated, so that the host side transmit and receive registers wrap around the same four 16-bit word addresses.

For example, host transmit register 0 on the HDI16 peripheral (address 0x04) can be obtained by the host by accessing any of the following memory mapped addresses: 0x20, 0x28, 0x30, or 0x38. This is critical for burst accesses as the source or destination addresses increment after each 16-bit access to the interface for all 16 transactions within that burst. By using the addressing as defined, if the first access is at 0x20, the last will be at 0x3e but, more importantly, the four host Tx/Rx registers will have been looped around four times.

13.5.6.1.4 Host 60x Bus: HDI16 Peripheral Interface Hardware Timings

The host UPM-controlled local bus and the HDI16 MSC8101 HDI16 host interface are both programmable. Careful programming of the host chip select registers and UPM can meet the HDI16 MSC8101 host port timings.

On any bus access the critical timing for both read and write is typically around the data latch point. For the UPM based read access, the host has the flexibility to latch data on a rising or falling LCLK edge. The falling LCLK edge is used here to latch the HDI16 data into the host MSC8101 at its earliest convenience.

After the data is latched, appropriate HDI16 port data hold time is ensured before the data strobe (DS) and chip select (CS1) are negated.

On a UPM write cycle, the critical action is in enveloping the DS assertion with CS asserted to ensure proper write data hold time after latching by the HDI16 host port.

Special attention needs to be given to both the host read and write access strobe (DS) negation times ($\overline{\text{HDS}}$ assert).

The HDI16 MSC8101 specifies some restrictions for consecutive register access, which results in a hold off negation time for the read and write access strobes.

Rather than restrict the firmware to avoid consecutive bus accesses to host port registers, the negation hold off times should be accommodated in the UPM hardware interface settings. Additional clocks must be built into the end of UPM based cycle giving appropriate time before the next bus cycle starts.

The timings can be readily adapted to allow external decode logic to be added to support chip selects for a larger number of DSP HDI16 host ports.

13.5.6.2 Interfacing to MSC8102 DSI

The MSC8102 direct-slave interface (DSI) gives an external host direct access to the MSC8102. 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 MSC8102 internal bus clock.

The DSI supports a 32- or 64-bit data bus. For connection to the local bus the DSI has to be configured in 32-bit mode. This is achieved through the DSP reset configuration.

The DSI supports two addressing modes, which are determined during the MSC8102 boot sequence. Refer to details in the MSC8102 documentation.

- Full address bus mode with HA[11:29] used in both 32-bit data mode and 64-bit data mode
- Sliding window mode with HA[14:29] used in both 32-bit data mode and 64-bit data mode

13.5.6.2.1 DSI in Asynchronous SRAM-Like Mode

The local bus supports the DSI single strobe as well as the DSI double strobes of operation. As an example the dual strobe configuration is shown below.

Figure 13-82 shows the interface to the MSC8102 DSI for asynchronous mode.

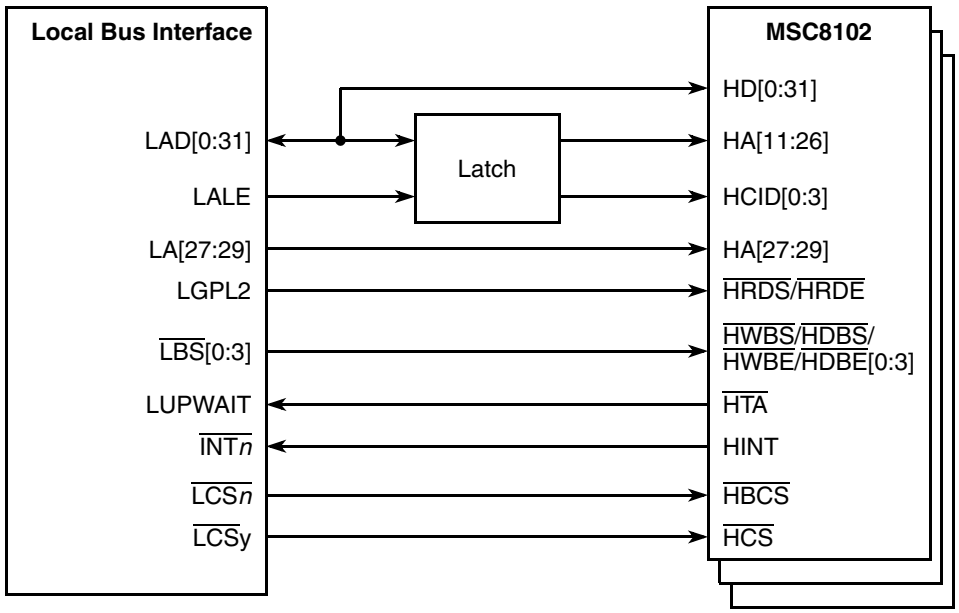


Figure 13-82. Interface to MSC8102 DSI in Asynchronous Mode

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 MSC8102. To allow 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 13-83 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 ($\overline{\text{HWBS}}$) on which the host chip select signal ($\overline{\text{HCS}}$) is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. The DSI will signal with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$), whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating $\overline{\text{HWBS}}$. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. The UWPL bit in the M*x*MR must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. The DSI samples the host address bus (HA) and the host data bus (HD) on the rising edge of $\overline{\text{HWBS}}$. In addition the assertion of $\overline{\text{HWBS}}$[0:3] are sampled at the end and are part of the access attributes.

Because the UPM is used for this mode, the DCR[4]:HTAAD should be set to 1 and DCR[9–10]:HTADT should be defined to a value different than 00. This mode is to be used in 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 MSC8102, then to prevent contention on the $\overline{\text{HTA}}$ signal, 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 MSC8102, the host must not start consecutive accesses before $\overline{\text{HTA}}$ is actively driven to a value of 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.

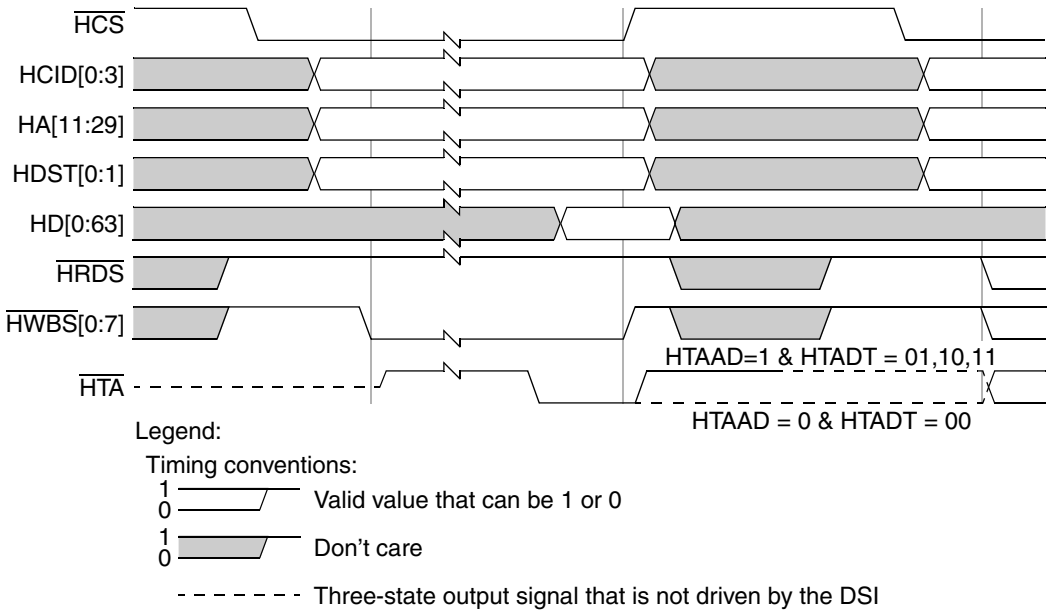


Figure 13-83. Asynchronous Write to MSC8102 DSI

Figure 13-84 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[8]: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. The DSI signals (with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$)) that data is valid, and 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. M*x*MR[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[4]:HTAAD should be set to 1 and the drive time control field, DCR[9–10]:HTADT, should be defined to a value different than 00. 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 the $\overline{\text{HCS}}$ signal between accesses. If the next access is not to the same MSC8102, 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 MSC8102, 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.

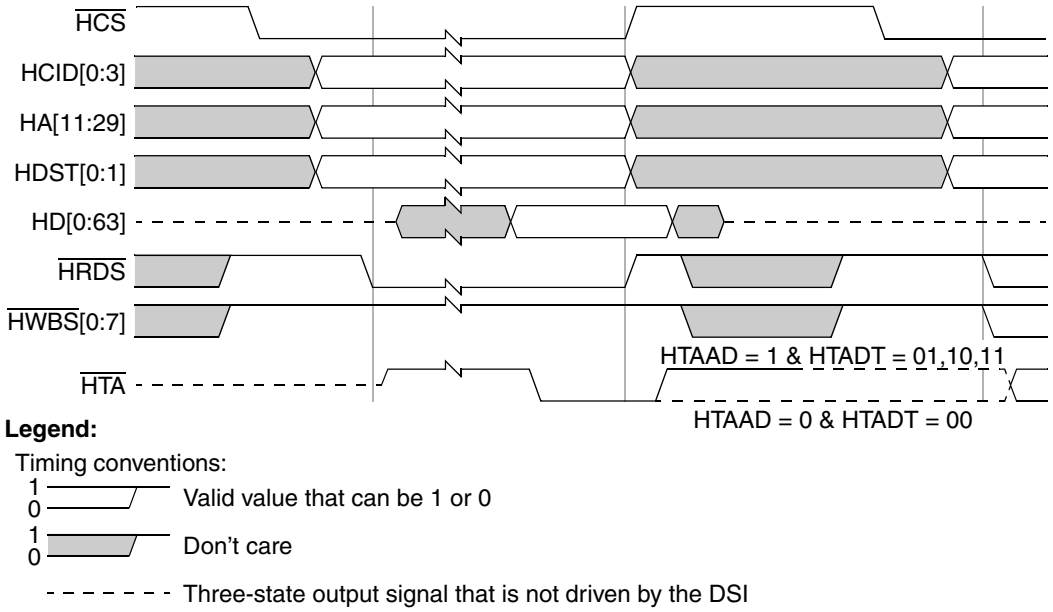


Figure 13-84. Asynchronous Read from MSC8102 DSI

13.5.6.2.2 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 MSC8102. This will optimize the bus utilization, especially if several MSC8102s are connected to one local bus. The UPM machine of the LBC must be used to implement this interface.

Figure 13-85 shows the interface for synchronous mode. Because the DSI will assert and negate $\overline{\text{HTA}}$ 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.

$\overline{\text{HTA}}$ drives LUPWAIT of the UPM. M*x*MR[UWPL] must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. Because this signal influences the internal state machine of the local bus clock, the local bus cannot react to $\overline{\text{HTA}}$ changes correctly within one local bus clock. Refer to Section 13.4.4.4.10, "Wait Mechanism (WAEN)," for more detailed information.

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 1×, 2×, 3×, or 4×, which should be sufficient for any divider ratio that would be used in this case.

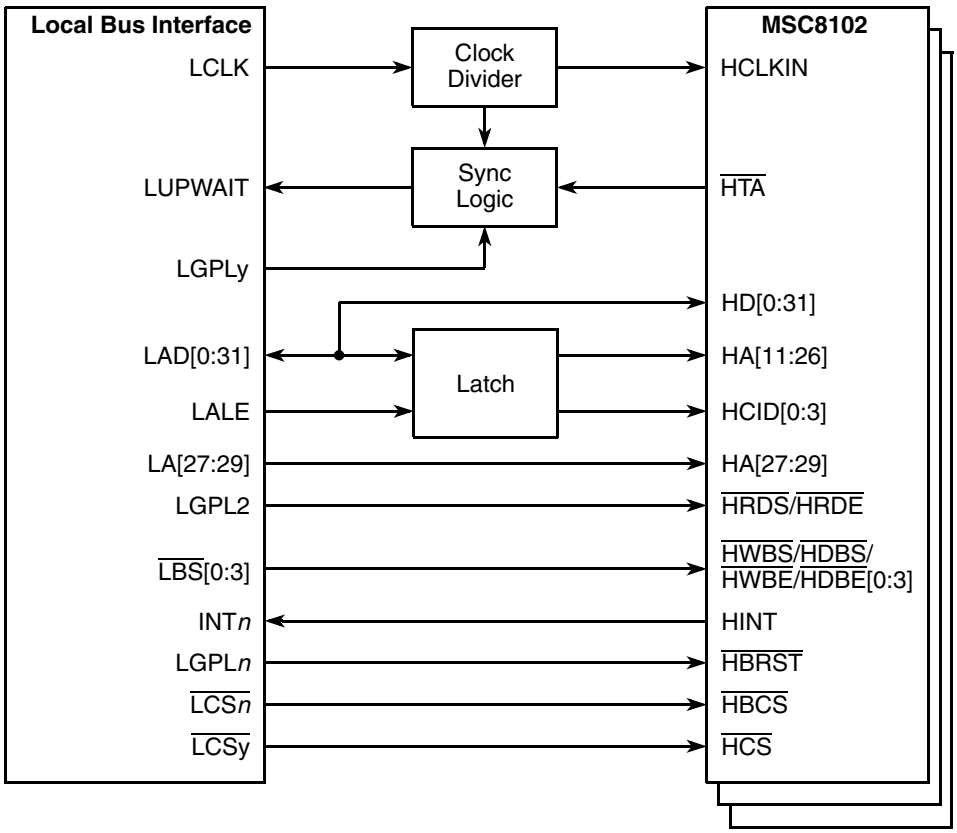


Figure 13-85. Interface to MSC8102 DSI in Synchronous Mode

This solution allows the local bus to react within multiple local bus clocks to the HTA signal and be still within one DSI clock. Typically, LUPWAIT is synchronized internally, and only 2 clocks after LUPWAIT changes, new data can be sampled or presented. For example, if the local bus clock ratio is 3× 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 2× the DSI clock, there is a special mode, where the LUPWAIT is NOT synchronized. Refer to Section 13.4.4.5, "Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge," for more detailed information. In this mode, data is sampled in the second sub-clock, which is the second half of the DSI clock. AC timing of LUPWAIT must be met in this mode; otherwise indeterministic behavior may occur.

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 cycle of the DSI, a mechanism must be used to ensure that the UPM changes transitions in a way that is synchronized to the DSI clock. The solution is to use a special synchronize cycle at the beginning of the pattern. A GPL signal is used to control a multiplexer and to activate 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 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 (GPL[0:4] are 1 when inactive, GPL5 is 0 when inactive) to start the synchronization process.

Figure 13-86 shows an example for 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 in length from one to n (clock ratio) local bus clocks.

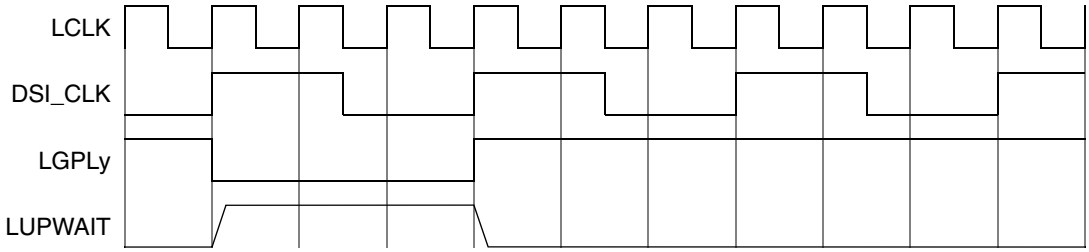


Figure 13-86. UPM Synchronization Cycle

The second column (compensation cycle) of Table 13-47 is intended to compensate for the reaction time of LUPWAIT to get in lockstep 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.

Table 13-47. UPM Synchronization Cycles

	Sync 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.

Synchronous Single Write

Figure 13-87 shows a synchronous single write access.

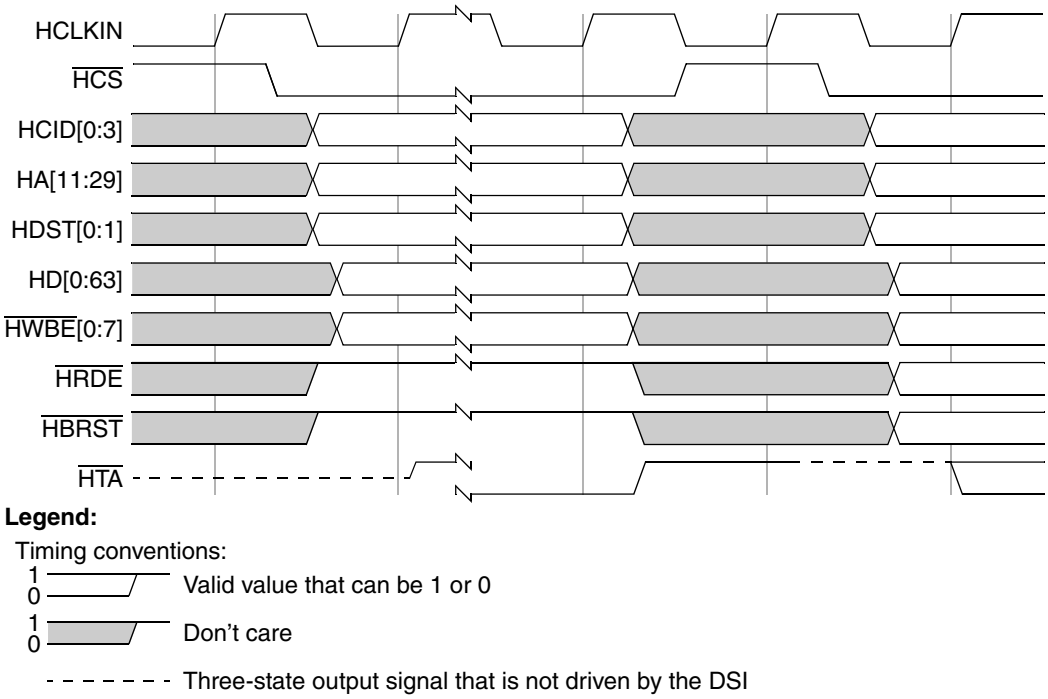


Figure 13-87. Synchronous Single Write to MSC8102 DSI

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 MSC8102 immediately on the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8102, 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.

Synchronous Single Read

Figure 13-88 shows a synchronous single read access.

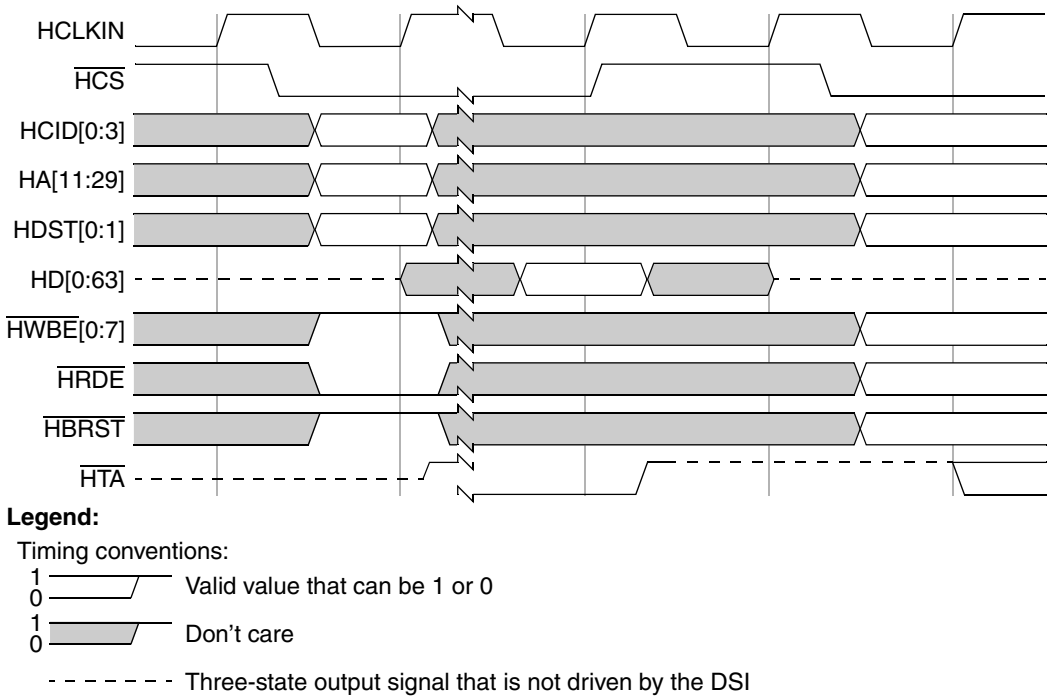


Figure 13-88. Synchronous Single Read from MSC8102 DSI

The DSI samples HA, HDST, HCID, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. HRDE is asserted, and HWBE and HBRST are negated. If DCR[8]:RPE is set (see MSC8102 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 HTA indicates that data is valid and the host must sample the HD and terminate the access. Because HTA is connected to the UPM LUPWAIT signal, all local bus signals are frozen until HTA goes to 0; then the UPM continues in its pattern. 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 MSC8102 immediately in the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8102, 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.

Synchronous Burst Write

Figure 13-89 shows a synchronous burst write access.

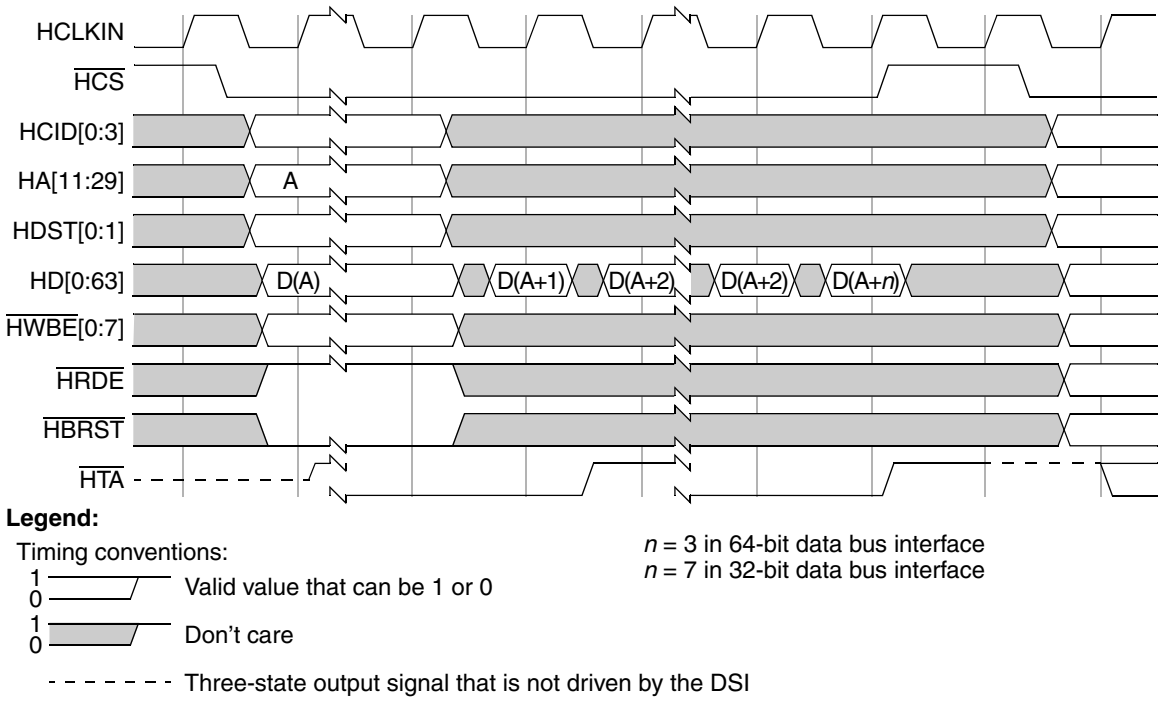


Figure 13-89. Synchronous Burst Write to MSC8102 DSI

The DSI samples HA, HDST, HCID, HD, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, 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. $\overline{\text{HWBE}}$ are asserted, $\overline{\text{HBRST}}$ is asserted, and $\overline{\text{HRDE}}$ is negated. Assertion of $\overline{\text{HTA}}$ 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 $\overline{\text{HTA}}$ is asserted immediately for each beat of the access. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. 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. After the last beat of the access, $\overline{\text{HTA}}$ 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 MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then, 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 to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

Synchronous Burst Read

Figure 13-90 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.

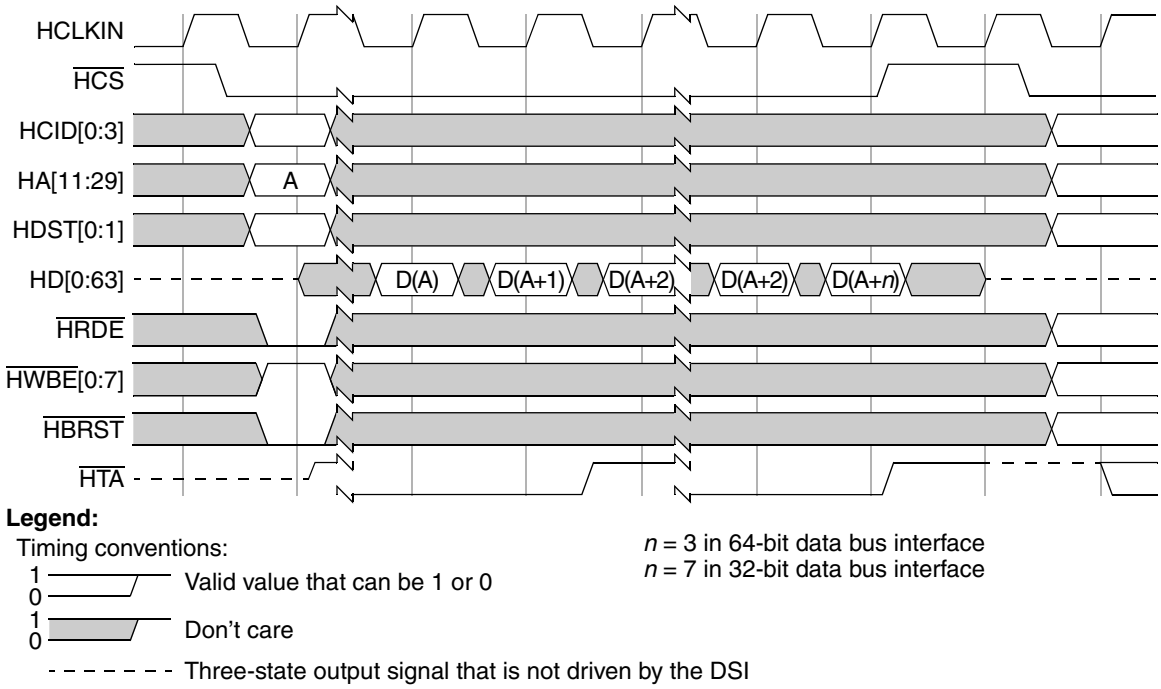


Figure 13-90. Synchronous Burst Read from MSC8102 DSI

$\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE}}$ are negated. When the DCR[8]:RPE bit (see the MSC8102 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 MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving the $\overline{\text{HTA}}$ signal. 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.

13.5.6.2.3 Broadcast Accesses

Using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8102 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 DSI error register (DER) OVF bit if there is an overflow during broadcast accesses. This bit can be cleared by writing a value of 1 to it.

NOTE

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[0]: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[0]: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 prior to the first broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In asynchronous mode, $\overline{\text{HWBS}}$[0:3]/$\overline{\text{HDBS}}$[0:3] assertion time should be at least the minimum, which is defined in the AC characteristics section of the *MSC8102 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.

13.5.6.3 Interfacing to EHPI from Texas Instruments TMS320C*xxxx* DSPs

The enhanced host port interface (EHPI) on DSPs from Texas Instruments provides a 16-bit-wide parallel port through which a host processor can directly access the memory of the DSP. The host and the DSP can exchange information through memory internal or external to the DSP and within the address reach of the EHPI. The EHPI uses 23-bit addresses, where each address is assigned to a 16-bit word in memory.

The EHPI has two modes, one for multiplexed address/data and one with separate buses.To allow the connection of multiple DSPs and other peripherals on the local bus, the use of the EHPI non-multiplexed mode is recommended. The EHPI uses the signals in Figure 13-48.

Table 13-48. EHPI Signals

Signals	Type	Description	Connect With
HD[15:0]	I/O/Z	Host data bus: Non-muxed mode: data only	LAD[0:15]
HA[19:4]	I	Host address bus: Non-muxed mode: addresses	Latched A[11:26]
HA[3:0]		Host address bus: lsbs	LA[27:30]
HBE[1:0]	I	Host byte-enable signals 00 Word, 0 MSB, 10 LSB, 11 Reserved	$\overline{\text{LBS}}$[0:1]
HCS	I	Chip select signal	$\overline{\text{LCS}n}$
HR/$\overline{\text{W}}$	I	R/$\overline{\text{W}}$ signal	LGPLy or inverted LBCTL
$\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$	I	Data strobe signals must be at least 2 DSP clocks wide • Host has separate active-low read and write strobe signals: connect one to $\overline{\text{HDS1}}$, one to $\overline{\text{HDS2}}$ • Host has one active-low strobe signal: connect to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, the other to 1 • Host has one active high-strobe signal: connect to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, the other to 0	LGPL*n*
HRDY	O	EHPI ready signal	LUPWAIT
HCNTL0	I	EHPI control signals. Non-muxed mode: • HCNTL = 1: access DSP data memory • HCNTL = 0: access EHPI control register	Application specific: either GPIO signal or latched address lines
HAS	I	Address strobe signal	
HMODE	I	EHPI mode signal High: non-muxed mode Low: muxed mode	High
RST_MODE	I	Reset mode signal	
HINT	O	DSP to host interrupt signal	INT*n*

To achieve the timings required by the DSP host port and optimize the bus usage, the use of one of the UPMs is recommended. Note that the DSP address signals reflect 16-bit addresses, whereas local bus address signals reflect byte addresses. This essentially shifts address signals by 1 bit.

The EHPI host port's two strobe signals, $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$, allow different options to control transfers. The UPM supports any of those options; however, the easiest is to use the single active-low strobe mode and connect one UPM LGPL signal (whatever is available) to $\overline{\text{HDS}}$.

In any case, the host port requires a HR/$\overline{\text{W}}$ signal. This can be generated by using another LGPL signal and allows to adopt the timing virtually without restrictions. Alternatively the designer can invert LBCTL to generate this signal. It is the responsibility of the UPM pattern designer to plan for the additional delay of that inverter in the UPM pattern to satisfy AC timings at the DSP host port.

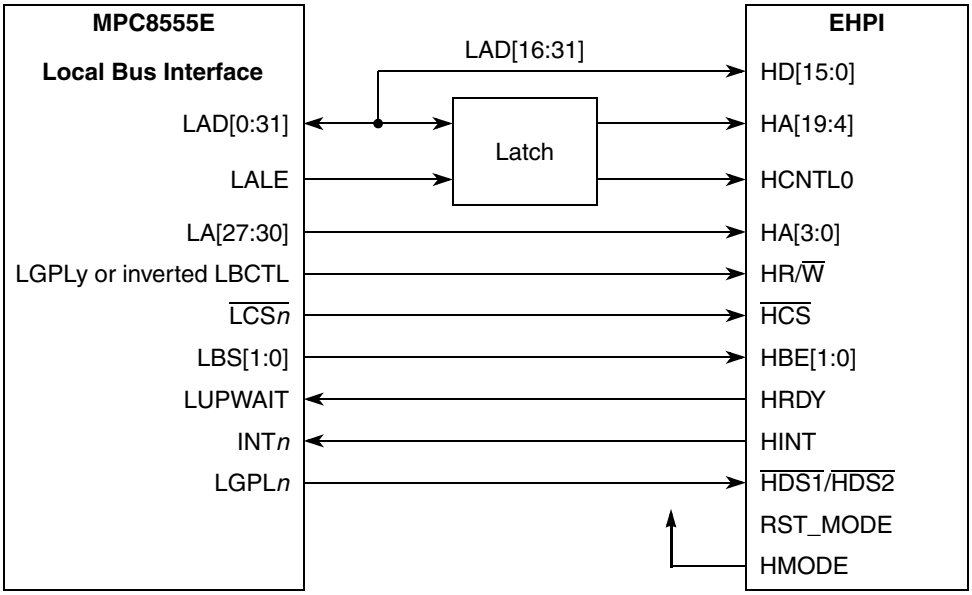
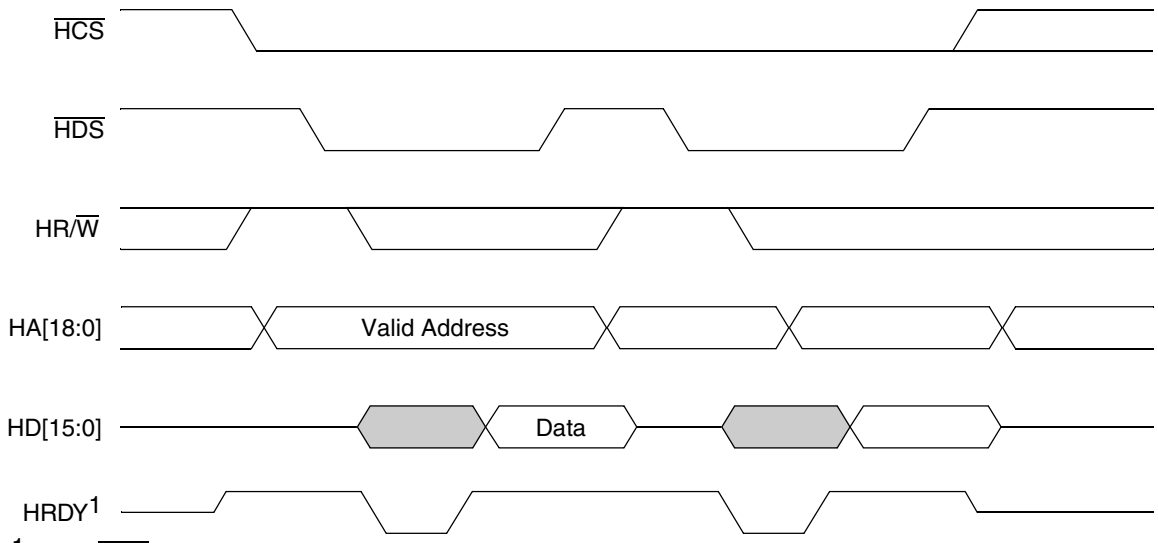


Figure 13-91. Interface to Texas Instruments EHPI in Non-Multiplexed Mode

The DSP does not necessarily have deterministic access times. HRDY indicates whether the EHPI is ready for an access. If the signal is low, wait states must be inserted in the cycle. The LUPWAIT function of the UPM provides this mechanism. M*x*MR[UWPL] must be set to connect to this active-low signal.

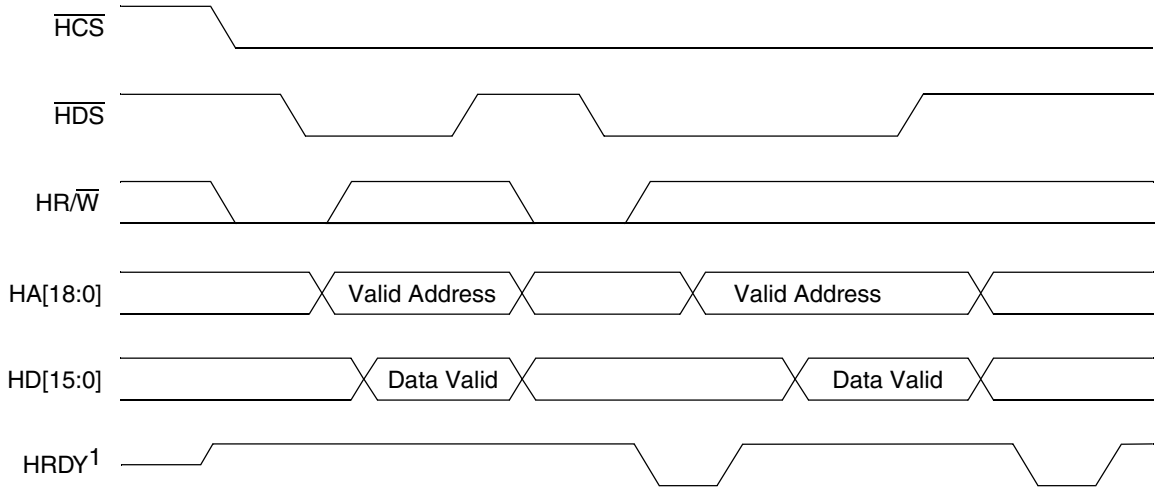
Figure 13-92 shows read timing required by the EHPI in non-multiplexed mode.



1 While HCS is not selected, HRDY is in Hi-Z state.

Figure 13-92. EHPI Non-Multiplexed Read Timings

Figure 13-93 shows write timing required by the EHPI in non-multiplexed mode.



1 While HCS is not selected, HRDY is in Hi-Z state.

Figure 13-93. EHPI Non-Multiplexed Write Timings

13.5.6.3.1 Expansion to Multiple DSPs

The connection shown above can be adapted easily to interface to multiple DSPs instead of only one. Each DSP host port needs to receive its own chip select and interrupt signals, and HRDY must be connected differently, depending on which DSP is used. All other signals can be connected to all host ports in parallel.

HRDY signals can be bussed for certain DSPs (such as, TMS320VC5510); for others (such as, TMS320VC5509), an external multiplexer must be used to decide which HRDY signal is routed to the local bus. This multiplexer can be controlled by the respective chip selects.

For a larger number of DSPs, this scheme must be extended with external logic, which uses additional address signals to generate multiple DSP HCSs for one local bus chip select and to multiplex the HINT and HRDY signals. The flexibility of the UPM allows for additional delay for that external logic.

Chapter 14
Three-Speed Ethernet Controllers

This chapter describes the two three-speed Ethernet controllers of the MPC8555E. The two controllers are referenced as TSEC1 and TSEC2.

14.1 Introduction

The Ethernet IEEE 802.3 standard protocol is widely used on LANs that are based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 standard protocols are similar and can coexist on the same LAN, both are referred to as Ethernet in this manual, unless otherwise noted. 10/100 Ethernet provides increased Ethernet speed from 10 to 100 megabits per second (Mbps) and provides a simple, cost-effective option for backbone and server connectivity. This three-speed Ethernet controller (TSEC) also implements a gigabit Ethernet protocol, which builds on top of the Ethernet protocol, but increases speed tenfold over 10/100 Ethernet to 1000 Mbps, or one gigabit per second (Gbps).

Gigabit Ethernet looks identical to Ethernet from the data link layer upward but it uses the ANSI X3T11 FiberChannel FC-0 (interface and media) and FC-1 (encode/decode) for the PHY Layer. In this manner, the standard takes advantage of the existing high-speed physical interface technology of FiberChannel while maintaining the IEEE 802.3 standard Ethernet frame format, backward compatibility for installed media, and the use of full- or half-duplex CSMA/CD.

The Ethernet protocol implements the bottom two layers of the open systems interconnection (OSI) 7-layer model, that is, the data link and physical sublayers. Figure 14-1 shows the typical Ethernet protocol stack and the relationship to the OSI model.

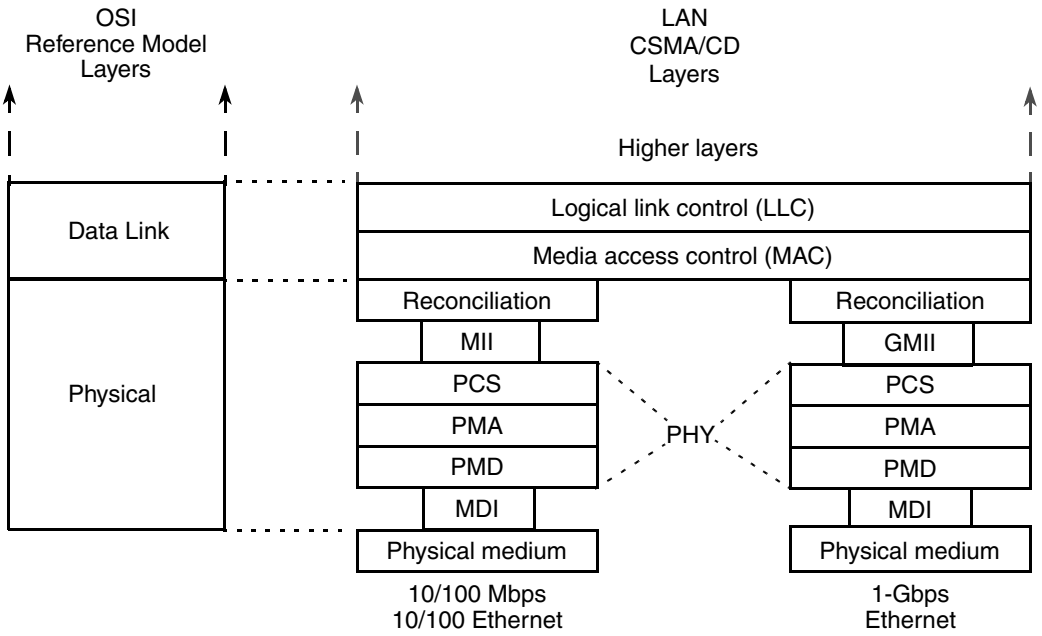


Figure 14-1. Ethernet Protocol in Relation to the OSI Protocol Stack

Gigabit Ethernet provides the following sublayers:

- Media access control (MAC) sublayer—The MAC sublayer provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.
- Reconciliation sublayer—The reconciliation sublayer acts as a command translator. It maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.
- MII (media-independent interface) sublayer—The MII sublayer provides a standard interface between the MAC layer and the physical layer for 10/100 Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.
- GMII (gigabit media-independent interface) sublayer—The GMII sublayer provides a standard interface between the MAC layer and the physical layer for 1-Gbps operation. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.
- PCS (physical coding sublayer)—The PCS sublayer is responsible for encoding and decoding data stream to and from the MAC sublayer. Medium (1000BASEX) 8B/10B coding is used for fiber. Medium (1000BASET) 8B1Q coding is used for unshielded twisted pair (UTP).
- PMA (physical medium attachment) sublayer—The PMA sublayer is responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices (SerDes) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers. For fiber medium (1000BASEX) the interface on the PMD side of the PMA is a 1-bit 1250-MHz signal, while on the PMA's PCS side the interface is a 10-bit interface (TBI) at 125 MHz. The TBI is an alternative to the GMII interface. If the TBI is

used, the gigabit Ethernet controller must be capable of performing the PCS function. For UTP medium, the PMD interface side of the PMA consists of four pair of 62.5-MHz PAM5 encoded signals, while the PCS side provides the 1250 Mbps input to a 8B1Q4 PCS.

- PMD (physical medium dependent) sublayer—The PMD sublayer is responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.
- MDI (medium-dependent interface) sublayer—MDI is a connector. It defines different connector types for different physical media and PMD devices.

Figure 14-2 describes the different physical interface standards.

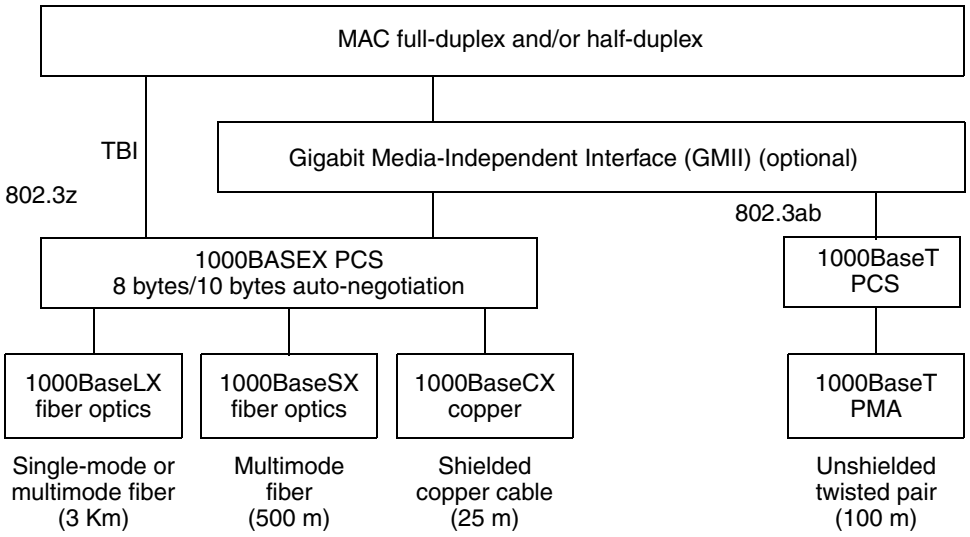


Figure 14-2. IEEE 802.3z and 802.3ab Physical Standards

Ethernet/IEEE 802.3 standard frames are based on the frame structure shown in Figure 14-3. The term ‘packet’ is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

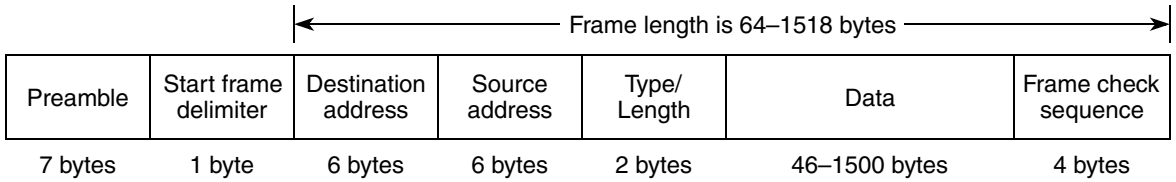


Figure 14-3. Ethernet/IEEE 802.3 Standard Frame Structure

The elements of an Ethernet frame are as follows:

- Preamble— The 7-byte preamble of alternating ones and zeros used for receiver timing synchronization (each byte containing the value 0x55).
- Start frame delimiter (SFD)—A sequence of 0xD5 (10101011 because the bit ordering is lsb first) indicates the beginning of the frame.
- 48-bit destination address (DA)—The first bit identifies the address as an individual address (0) or a group address (1). The second bit is used to indicate whether the address is locally-defined (1) or globally-defined (0).

- 48-bit source address (SA)— (Original versions of the IEEE 802.3 standard allowed 16-bit addressing, which has never been used widely.)
- Ethernet type field/IEEE 802.3 standard length field—The type field signifies the protocol (for example, TCP/IP) used in the rest of the frame. The length field specifies the length of the data portion of the frame. For both Ethernet and IEEE 802.3 standard frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This limitation requires that a type field be identified by a decimal number equal to or greater than 1536 (0x0600) but less than 65535 (0xFFFF). If the number, however, is between 0 and 1,500 (0x0000 through 0x05DC) then this field indicates the length of the MAC client data. The range from 1,501 to 1,535 (0x5DD through 0x5FF) was intentionally left undefined.
- Data and padding—Padding is optional. It is only needed if the data is smaller than 46 octets (one octet = one byte) to ensure the minimum frame size of 64 octets as specified in the IEEE 802.3 standard. In the 802.3x standard, the first two octets of the data field are used as opcode (OP) (pause = 0x0001) and the second two octets are used to transmit a pause time (PT) parameter (pausetime = 0x0000 for on and 0xFFFF for off). In addition, a third two-octet field can be used for an extended pause control parameter (PTE). Because the use of these fields varies with the protocol used, the ability to examine them and report their content can significantly accelerate Ethernet frame processing.
- Frame-check sequence (FCS)—Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD and CRC.

Figure 14-4 provides additional details of the Ethernet/IEEE 802.3 standard frame structure.

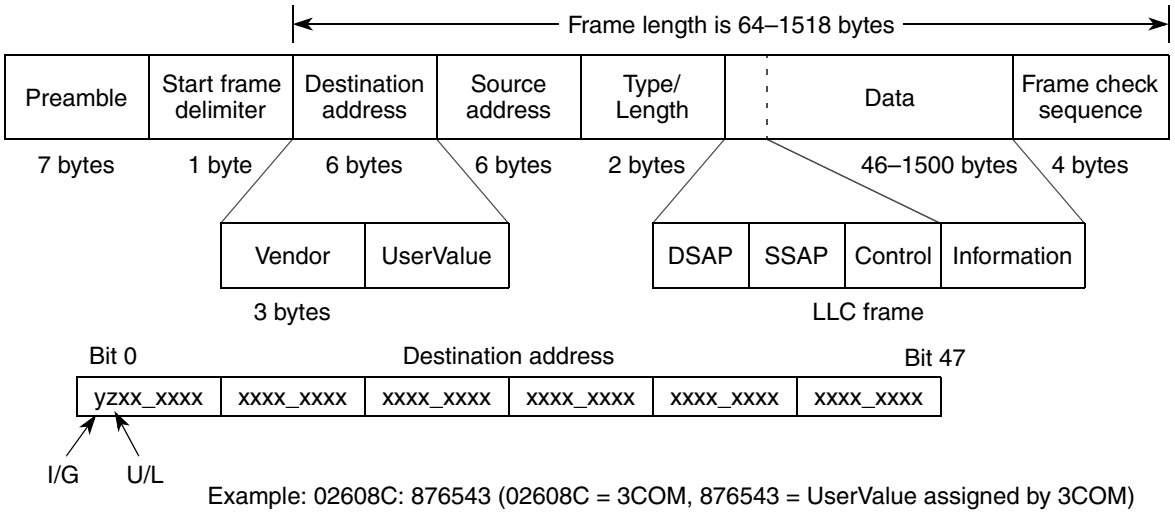


Figure 14-4. Ethernet/IEEE 802.3 Standard Frame Structure With More Details

Relative to Figure 14-4, the IEEE 802.3 standard, section 3.11 (MAC frame format) defines the frame format such that the octets of a frame are transmitted from left to right (preamble first, the FCS last), and the bits of each octet are transmitted least-significant bit (lsb) first. The destination address example shown in Figure 14-4 (02608C:876543) which would normally be written as:

0000 0010 0110 0000 1000 1100 1000 0111 0110 0101 0100 0011

is transmitted bit by bit as:

0100 0000 0000 0110 0011 0001 1110 0001 1010 0110 1100 0010

which is an individual address (because the lsb is cleared) but locally-defined (because the second least-significant bit is set).

When first originated, a type field was used for protocol identification. The IEEE 802.3 standard eliminated the type field, replacing it with the length field. The length field is used to identify the length, in bytes, of the data field. The protocol type in the IEEE 802.3 standard frames are held within the data portion of the packet. The logical link control (LLC) is responsible for providing services to the network layer regardless of media type, such as FDDI, Ethernet, token ring, and others. The LLC layer makes use of LLC protocol data units (PDUs) in order to communicate between the media access control (MAC) layer and the upper layers of the protocol stack. Three variables determine access into the upper layers through the LLC-PDU.

The variables include the destination service access point (DSAP), the source service access point (SSAP), and a control variable. The DSAP address specifies a unique identifier within the station providing protocol information for the upper layer. The SSAP provides the same information for the source address.

The LLC defines service access for protocols that conform to the open system interconnection (OSI) model for network protocols. However, many protocols do not obey the rules for those layers and additional information must be added to the LLC in order to provide information regarding those protocols. Protocols that fall into this category include IP and IPX. The method used to provide this additional protocol information is called a subnetwork access protocol (SNAP) frame. A SNAP encapsulation is indicated by the DSAP and SSAP addresses being set to 0xAA and the LLC control field being set to 0x03. If that address is seen, a SNAP header follows. The SNAP header is five bytes long. The first three bytes consist of the organization code (SNAP OUI), which is assigned by the IEEE. The last two bytes become the type value set from the original Ethernet specifications if SNAP OUI = 0, or they become a SNAP protocol identifier if SNAP OUI is non zero.

The three-speed Ethernet controller (TSEC) allows the flexibility to accelerate the identification and retrieval of all the standard and non-standard protocols mentioned above. Figure 14-5 shows the block diagram of the TSEC.

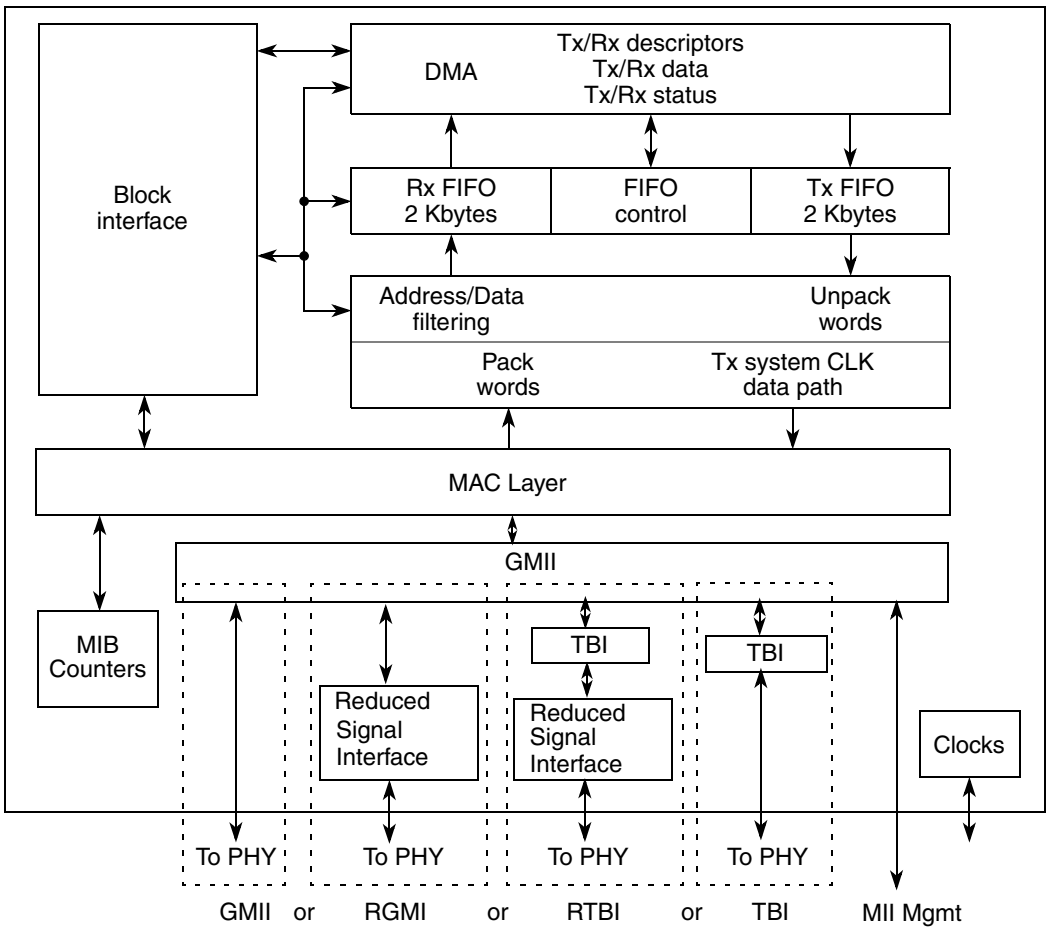


Figure 14-5. TSEC Block Diagram

14.1.1 Three-Speed Ethernet Controller Overview

The TSEC is designed to support 10-, 100-, and 1000-Mbps Ethernet/IEEE 802.3 standard networks and contains the following components:

- Ethernet media access controller (MAC)
- First-in first-out (FIFO) controller
- Direct memory access (DMA) controller
- Ten-bit interface (TBI)
- Register-based statistical module that supports management information base (MIB) remote monitoring (RMON)

The most-significant byte of data in a receive or transmit data buffer corresponds to the most-significant byte of a frame, respectively.

The complete TSEC is designed for single MAC applications. The TSEC supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver:

- MII interface running at 10/100 Mbps
- GMII interface running at 1 Gbps
- TBI interface that can be connected to a SerDes device for fibre channel applications.
- Reduced signal count versions of the GMII (RGMII) and ten-bit (RTBI) interfaces

While most of this document refers to the non-reduced signal count interfaces, it must be understood that these references also apply to the reduced signal count interfaces.

The TSEC software programming model is similar to the MPC8260 (PowerQUICC II) device. Hence, it enables Freescale customers to leverage already implemented Ethernet drivers, reducing the software development cycle.

14.2 Features

The MPC8555E TSEC includes these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant
- Support for different Ethernet physical interfaces:
  - 10/100/1Gb IEEE 802.3 GMII
  - 10/100 Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII
  - 1-Gbps IEEE 802.3z TBI
  - 10/100 Mbps RGMII
  - 1-Gbps full-duplex RGMII
  - 1-Gbps RTBI
- Full- and half-duplex support (1 Gbps supports only full-duplex)
- IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)
- Support for out-of-sequence transmit queue (for initiating flow-control)
- Programmable maximum frame length supports jumbo frames (up to 9.6 Kbytes) and IEEE 802.1 virtual local area network (VLAN) frames
- Retransmission from transmit FIFO following a collision
- Support for CRC generation and verification of inbound/outbound packets
- Address recognition
  - Each exact match can be programmed to be accepted or rejected
  - Broadcast address (accept/reject)
  - Exact match 48-bit individual (unicast) address
  - Hash (256-bit hash) check of individual (unicast) addresses
  - Hash (256-bit hash) check of group (multicast) addresses

— Promiscuous mode

- Extraction data and its associated buffer descriptors can be directed to the processor's L2 cache to reduce access latency
- Interrupt coalescing subject to a threshold frame counter and/or a threshold timer (Independent functionality for received and transmitted frames)
- RMON statistics support

14.3 Modes of Operation

The primary TSEC operational modes are the following:

- Full- and half-duplex operation

  This is determined by MACCFG2 register's full-duplex bit. Full-duplex mode is intended for use on point to point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

  If configured in half-duplex mode (only 10- and 100-Mbps operation; MACCFG2 register's full-duplex bit is cleared), the MAC complies with the IEEE CSMA/CD access method.

  If configured in full-duplex mode (10/100/1Gb operation; MACCFG2 register's full-duplex bit is set), the MAC supports flow control. If flow control is enabled, it allows the MAC to receive or send PAUSE frames.
- 10- and 100-Mbps MII interface operation

  The MAC-PHY interface operates in MII mode by configuring bits MACCFG2[22–23] = 01. The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the TSEC*n*_TX_CLK and TSEC*n*_RX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed or it may be controlled by software through the serial management interface (EC_MDC/EC_MDIO signals) to the transceiver.
- 1-Gbps GMII and TBI interface operation

  The MAC-PHY interface operates in GMII mode by configuring bits MACCFG2[22–23] = 10. The GMII is the gigabit media-independent interface defined by the 802.3 standard for 1-Gbps operation.

  Independently, the MAC-PHY interface can also operate in TBI mode. Note that either the TBI or GMII interface is chosen; not both at the same time. TBI is the ten-bit interface which contains PCS functions (ten-bit encoding/decoding) as defined by the 802.3 standard.

  In reduced signal count mode (RGMII or RTBI), the MAC remains configured in GMII or TBI but the TSEC multiplexes and decodes the input signals and provides the MAC with the expected interface.

  TSEC provides the TSEC*n*_GTX_CLK to the PHY in either GMII or TBI mode of operation.
- Address recognition options

  The options supported are promiscuous, broadcast, exact individual address hash or exact match, and multicast hash match. For detailed descriptions refer to Section 14.5.3.8.1, "Individual Address Registers 0–7 (IADDRn)," Section 14.5.3.8.2, "Group Address Registers 0–7

(GADDRn)," Section 14.5.3.4.1, "Receive Control Register (RCTRL)," and Section 14.6.2.6, "Frame Recognition."

- RMON support

  See Section 14.6.2.5, "RMON Support."

- Internal loopback

  Internal loopback mode is selected through the loopback bit in the MACCFG1 register. See Section 14.6.2.10, "Internal and External Loopback," for details.

14.4 External Signals Description

This section defines the TSEC signals. The buses are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention (that is, TxD[7:0] means 0 is the lsb). Notice that except for external physical interfaces the buses and registers follow a big-endian format.

The TSEC network interface supports multiple options:

- The MII option requires 18 I/O signals (including the EC_MDIO and EC_MDC MII management interface) and supports both a data and a management interface to the PHY (transceiver) device. The MII option supports both 10- and 100-Mbps Ethernet rates.
- The GMII option is a superset of the MII signals and supports a 1-Gbps Ethernet rate.
- The TBI interface shares signals with the GMII interface signals.
- Finally, RGMII and RTBI options are reduced-signal implementations of the GMII and TBI interfaces.

14.4.1 Detailed Signal Descriptions

Table 14-1 contains detailed descriptions of the TSEC interface signals. For RGMII mode details please refer to the Hewlett-Packard reduced gigabit media-independent interface (RGMII) specification version 1.2a, dated 9/22/2000. All other modes follow the IEEE 802.3 standard, 2000 Edition. Input signals not used are internally disabled. Output signals not used are driven low.

Table 14-1. TSEC Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TSEC*n*_COL	I	Collision input. The behavior of this signal is not specified while in full-duplex mode.	
		State Meaning	Asserted/Negated—In MII mode, this signal is asserted upon detection of a collision, and must remain asserted while the collision persists. This signal is not used in the TSEC GMII, TBI, RTBI and RGMII modes.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
TSEC*n*_CRS	I	Carrier sense input. In TBI and RTBI modes this signal can be used as SDET (signal detect), an optional signal that some PHYs generate in TBI or RTBI modes. This signal is not used in the TSEC GMII or RGMII modes.	
		State Meaning	Asserted/Negated—In MII mode, TSEC*n*_TX_CLK is asserted while the transmit or receive medium is not idle. In the event of a collision, TSEC*n*_CRS must remain asserted for the duration of the collision.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
TSEC*n*_GTX_CLK	O	Gigabit transmit clock. This signal is an output from the TSEC into the PHY. In GMII, TBI, or RTBI mode TSEC*n*_GTX_CLK is a 125-MHz clock that provides a timing reference for TX_EN, TXD, and TX_ER. In RGMII mode TSEC*n*_GTX_CLK becomes the transmit clock and provides timing reference during 1000Base-T (125-MHz), 100Base-T (25-MHz) and 10Base-T (2.5-MHz) transmissions. This signal is not used in MII mode.	
EC_GTX_CLK125	I	Gigabit transmit 125-MHz source. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY. In GMII, RGMII, RTBI or TBI mode, EC_GTX_CLK125 is a 125-MHz input into the TSEC and is used to generate all 125-MHz related signals and clocks. This input is not used in MII mode.	
EC_MDC	O	Management data clock. In GMII, MII, RGMII, RTBI or TBI mode this signal is a clock (typically 2.5 MHz) supplied by the MAC (IEEE–set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed). This clock is generated by dividing the core complex bus (CCB) clock by eight. The ratio can be modified further by writing to MIIMCFG[29:31]. Note that this signal is used during reset to configure the TSEC interface to ‘reduced-signal’ or ‘non-reduced-signal’ mode. See Section 4.4.3, “Power-On Reset Configuration,” for more information.	
EC_MDIO	I/O	Management data input/output, BIDI	
		State Meaning	Asserted/Negated—In GMII, MII, RGMII, RTBI or TBI mode EC_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles.
		Timing	Asserted/Negated—This signal is required to be synchronous with the EC_MDC signal.
TSEC*n*_RX_CLK	I	Receive clock. In GMII, MII or RGMII mode, the receive clock TSEC*n*_RX_CLK is a continuous clock (2.5, 25, or 125 MHz) that provides a timing reference for TSEC*n*_RX_DV, TSEC*n*_RXD, and TSEC*n*_RX_ER. In TBI mode, TSEC*n*_RX_CLK is the input for a 62.5-MHz PMA receive clock, 0 split phase with PMA_RX_CLK1 and is supplied by the SerDes. In RTBI mode it is a 125-MHz receive clock.	

Table 14-1. TSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_RX_DV	I	Receive data valid. In GMII or MII mode, if TSEC*n*_RX_DV is asserted, the PHY is indicating that valid data is present on the GMII and MII interfaces. In RGMII mode, TSEC*n*_RX_DV becomes RX_CTL. The RX_DV and RX_ERR are received on this signal on the rising and falling edges of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 8. Together, with RCG[9] and RCG[7:0], they represents the 10-bit encoded symbol of GMII receive signals. In RTBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 4 and 9. On the positive edge of the TSEC*n*_RX_CLK, RCG[4] and RCG[3:0] represents the first half of the 10-bit encoded symbol. On the negative edge of the TSEC*n*_RX_CLK, RCG[9] and RCG[8:4] represents the second half of the 10-bit encoded symbol.
TSEC*n*_ RXD[7:0]	I	Receive data in. In GMII mode, TSEC*n*_RXD[7:4] with TSEC*n*_RXD[3:0], represent one complete octet of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. In TBI mode, TSEC*n*_RXD[7:4] represents RCG[7:4]. Together, with RCG[9:8] and RCG[3:0], they represent the 10-bit encoded symbol of GMII receive signals. In GMII mode, TSEC*n*_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. A completely-formed SFD must be passed across the MII. While TSEC*n*_RX_DV is not asserted, TSEC*n*_RXD has no meaning. In RGMII or RTBI mode, TSEC*n*_RXD[3:0] are received on the rising edge of TSEC*n*_RX_CLK and TSEC*n*_RXD[7:4] are received on the falling edge of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RXD[3:0] represents RCG[3:0]. Together, with RCG[9:4], they represent the 10-bit encoded symbol of GMII receive signals.
TSEC*n*_RX_ER	I	Receive error **State Meaning**: Asserted/Negated—In GMII or MII mode, if TSEC*n*_RX_ER and TSEC*n*_RX_DV are asserted, the PHY has detected an error in the current frame. In TBI mode, TSEC*n*_RX_ER represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals. This signal is not used in the TSEC RTBI or RGMII modes.
TSEC*n*_TX_ CLK	I	Transmit clock in. In MII mode, TSEC*n*_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSEC*n*_TX_EN, TSEC*n*_TXD, and TSEC*n*_TX_ER signals. In GMII mode, TSEC*n*_TX_CLK provides the 2.5 or 25-MHz timing reference during 10Base-T and 100Base-T and comes from the PHY. In 1000Base-T this clock is not used and TSEC*n*_GTX_CLK (125 MHz) becomes the timing reference. The TSEC*n*_GTX_CLK is generated in the TSEC and provided to the PHY and the MAC. The TSEC*n*_TX_CLK is generated in the PHY and provided to the MAC. In TBI mode, this signal is PMA receive clock 1 62.5 MHz, split phase with PMA_RX_CLK0, and is supplied by the SerDes. This signal is not used in the TSEC RTBI or RGMII modes.

Table 14-1. TSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_ TXD[7:0]	O	Transmit data out. In GMII mode, TSEC*n*_TXD[7:4], together with TSEC*n*_TXD[3:0], represent one complete octet of data to be sent from the MAC to the PHY when TSEC_TX_DV is asserted and has no meaning while TSEC*n*_TX_EN is negated. In TBI mode, TSEC*n*_TXD[7:4] represents transmit code group (TCG) bits 7:4. Together, with TCG[9:8] and TCG[3:0], they represent the 10-bit encoded symbol. In GMII or MII mode, TSEC*n*_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when TSEC*n*_TX_EN is asserted and have no meaning while TSEC*n*_TX_EN is negated. In RGMII or RTBI mode, TSEC*n*_TXD[3:0] are transmitted on the rising edge of TSEC*n*_TX_CLK, and TSEC*n*_TXD[7:4] are transmitted on the falling edge of TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TXD[3:0] represents TCG[3:0]. Together, with TCG[9:4], they represent the 10-bit encoded symbol. Note that some of these signals are also used during reset to configure the TSEC interface in GMII or TBI mode. Additionally, some of these signals are used for other reset configuration settings for the MPC8555E. See Section 4.4.3, “Power-On Reset Configuration,” for more information.
TSEC*n*_TX_EN	O	Transmit data valid. In GMII or MII mode, if TSEC*n*_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII’s or the MII’s TSEC*n*_TXD signals. In RGMII mode, TSEC*n*_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this signal on rising and falling edges of the TSEC*n*_TX_CLK, respectively. In TBI mode, TSEC*n*_TX_EN represents TCG[8]. Together, with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol. In RTBI mode, TSEC*n*_TX_EN represents TCG[4]. Together with TCG[9], TCG[3:0] and TCG[8:5], they represent the 10-bit encoded symbol.
TSEC*n*_TX_ER	O	Transmit error. In GMII or MII mode, assertion of TSEC*n*_TX_ER for one or more clock cycles while TSEC*n*_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSEC*n*_TX_ER has no effect while operating at 10 Mbps or while TSEC*n*_TX_EN is negated. This signal transitions synchronously with respect to TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TX_ER represents TCG[9]. Together, with TCG[8:0], they represents the 10-bit encoded symbol. This signal is not used in the TSEC RTBI or RGMII modes.

14.5 Memory Map/Register Definition

The TSEC uses a software model similar to that employed by the fast Ethernet function supported on the Freescale MPC8260 CPM FCC and in the FEC of the MPC860T.

The TSEC device is programmed by a combination of control/status registers (CSR) and buffer descriptors. The CSRs are used for mode control, interrupts, and to access status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

All accesses to and from the registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.

This section of the document defines the memory map and describes the registers in detail. The buffer descriptor is described in Section 14.6.3, “Buffer Descriptors.”

The ten-bit interface (TBI) module MII registers are also described in this section. The TBI registers are defined like PHY registers and, as such, are accessed through the MII management interface in the same

way as the PHYs are accessed. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to Section 14.5.4, “Ten-Bit Interface (TBI).”

14.5.1 Top-Level Module Memory Map

The TSEC implementation requires 4 Kbytes of memory-mapped space, of which more than 1 Kbyte is reserved for future expansion. The space is divided into the following sections:

- General control/status registers
- Transmit-specific control/status registers
- Receive-specific control/status registers
- MAC registers
- Event/statistic counters held in the MIB block
- Hash function registers

Table 14-2 defines the top-level memory map.

Table 14-2. Module Memory Map Summary

Address	Function
000–0FF	TSEC general control/status registers
100–2FF	TSEC transmit control/status registers
300–4FF	TSEC receive control/status registers
500–5FF	TSEC MAC registers
600–7FF	TSEC RMON MIB registers
800–8FF	TSEC HASH function registers
900–AFF	Reserved
B00–BFF	TSEC attribute registers
C00–FFF	Future expansion space

14.5.2 Detailed Memory Map—Control/Status Registers

Table 14-3 lists the address, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for both TSECs. That is, TSEC1 starts at address offset 0x2_4000 and TSEC2 starts at address offset 0x2_5000. The registers for TSEC1 are listed in Table 14-3, but the registers for TSEC2 are not. Note that in Table 14-3 the registers are the same for TSEC2 except that the offset changes from 0x2_4*nnn* to 0x2_5*nnn*. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 14-3. Module Memory Map

Offset	Name	Access[1]	Reset	Section/Page
TSEC1 General Control and Status Registers				
0x2_4000–0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	14.5.3.1.1/14-19
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	14.5.3.1.2/14-22
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	14.5.3.1.3/14-24
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	14.5.3.1.4/14-25
0x2_4024	MINFLR—Minimum frame length register	R/W	0x0000_0040	14.5.3.1.5/14-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	14.5.3.1.6/14-26
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	14.5.3.1.7/14-27
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	14.5.3.1.8/14-28
0x2_4034–0x2_4088	Reserved	R	0x0000_0000	—
TSEC1 FIFO Control and Status Registers				
0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	14.5.3.2.1/14-30
0x2_408C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	14.5.3.2.2/14-30
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	14.5.3.2.3/14-31
0x2_409C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	14.5.3.2.4/14-31
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	14.5.3.3.1/14-32
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	14.5.3.3.2/14-33
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TBDLEN—TxBD data length register	R	0x0000_0000	14.5.3.3.3/14-34
0x2_4110	TXIC—Transmit interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.3.4/14-34
0x2_4114–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	14.5.3.3.5/14-35
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4184	TBPTR—TxBD pointer register	R/W	0x0000_0000	14.5.3.3.6/14-35
0x2_4188–0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TBASE—TxBD base address register	R/W	0x0000_0000	14.5.3.3.7/14-36
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	14.5.3.3.8/14-36
0x2_42B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	14.5.3.3.9/14-38
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	14.5.3.4.1/14-39
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	14.5.3.4.2/14-40
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RBDLEN—RxBD data length register	R	0x0000_0000	14.5.3.4.3/14-40
0x2_4310	RXIC—Receive interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.4.4/14-41
0x2_4314–0x2_4320	Reserved	R	0x0000_0000	—
0x2_4324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	14.5.3.4.5/14-42
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	14.5.3.4.6/14-42
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RBPTR—RxBD pointer register	R/W	0x0000_0000	14.5.3.4.7/14-43
0x2_4388–0x2_4400	Reserved	R	0x0000_0000	—
0x2_4404	RBASE—RxBD base address register	R/W	0x0000_0000	14.5.3.4.8/14-44
0x2_4408–0x2_44FC	Reserved	R	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	14.5.3.6.1/14-47
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	14.5.3.6.2/14-48
0x2_4508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	14.5.3.6.3/14-49
0x2_450C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	14.5.3.6.4/14-50

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	14.5.3.6.5/14-51
0x2_4514–0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	14.5.3.6.6/14-52
0x2_4524	MIIMCOM—MII Management command register	R/W	0x0000_0000	14.5.3.6.7/14-53
0x2_4528	MIIMADD—MII Management address register	R/W	0x0000_0000	14.5.3.6.8/14-53
0x2_452C	MIIMCON—MII Management control register	W	0x0000_0000	14.5.3.6.9/14-54
0x2_4530	MIIMSTAT—MII Management status register	R	0x0000_0000	14.5.3.6.10/14-55
0x2_4534	MIIMIND—MII Management indicator register	R	0x0000_0000	14.5.3.6.11/14-55
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	IFSTAT—Interface status register	R/W	0x0000_0000	14.5.3.6.12/14-56
0x2_4540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	14.5.3.6.13/14-56
0x2_4544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	14.5.3.6.14/14-57
0x2_4548–0x2_467C	Reserved	R	0x0000_0000	—
TSEC1 RMON MIB Registers				
TSEC1 Transmit and Receive Counters				
0x2_4680	TR64—Transmit and receive 64-byte frame counter register	R/W	0x0000_0000	14.5.3.7.1/14-58
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter register	R/W	0x0000_0000	14.5.3.7.2/14-58
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter register	R/W	0x0000_0000	14.5.3.7.3/14-59
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter register	R/W	0x0000_0000	14.5.3.7.4/14-59
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter register	R/W	0x0000_0000	14.5.3.7.5/14-60
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter register	R/W	0x0000_0000	14.5.3.7.6/14-60
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count register	R/W	0x0000_0000	14.5.3.7.7/14-61
TSEC1 Receive Counters				
0x2_469C	RBYT—receive byte counter register	R/W	0x0000_0000	14.5.3.7.8/14-61
0x2_46A0	RPKT—receive packet counter register	R/W	0x0000_0000	14.5.3.7.9/14-62
0x2_46A4	RFCS—receive FCS error counter register	R/W	0x0000_0000	14.5.3.7.10/14-62
0x2_46A8	RMCA—receive multicast packet counter register	R/W	0x0000_0000	14.5.3.7.11/14-63

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_46AC	RBCA—receive broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.12/14-63
0x2_46B0	RXCF—receive control frame packet counter register	R/W	0x0000_0000	14.5.3.7.13/14-64
0x2_46B4	RXPF—receive PAUSE frame packet counter register	R/W	0x0000_0000	14.5.3.7.14/14-64
0x2_46B8	RXUO—receive unknown OP code counter register	R/W	0x0000_0000	14.5.3.7.15/14-65
0x2_46BC	RALN—receive alignment error counter register	R/W	0x0000_0000	14.5.3.7.16/14-65
0x2_46C0	RFLR—receive frame length error counter register	R/W	0x0000_0000	14.5.3.7.17/14-66
0x2_46C4	RCDE—receive code error counter register	R/W	0x0000_0000	14.5.3.7.18/14-66
0x2_46C8	RCSE—receive carrier sense error counter register	R/W	0x0000_0000	14.5.3.7.19/14-67
0x2_46CC	RUND—receive undersize packet counter register	R/W	0x0000_0000	14.5.3.7.20/14-67
0x2_46D0	ROVR—receive oversize packet counter register	R/W	0x0000_0000	14.5.3.7.21/14-68
0x2_46D4	RFRG—receive fragments counter register	R/W	0x0000_0000	14.5.3.7.22/14-68
0x2_46D8	RJBR—receive jabber counter register	R/W	0x0000_0000	14.5.3.7.23/14-69
0x2_46DC	RDRP—receive drop register	R/W	0x0000_0000	14.5.3.7.24/14-69
TSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter register	R/W	0x0000_0000	14.5.3.7.25/14-70
0x2_46E4	TPKT—Transmit packet counter register	R/W	0x0000_0000	14.5.3.7.26/14-70
0x2_46E8	TMCA—Transmit multicast packet counter register	R/W	0x0000_0000	14.5.3.7.27/14-71
0x2_46EC	TBCA—Transmit broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.28/14-71
0x2_46F0	TXPF—Transmit PAUSE control frame counter register	R/W	0x0000_0000	14.5.3.7.29/14-72
0x2_46F4	TDFR—Transmit deferral packet counter register	R/W	0x0000_0000	14.5.3.7.30/14-72
0x2_46F8	TEDF—Transmit excessive deferral packet counter register	R/W	0x0000_0000	14.5.3.7.31/14-73
0x2_46FC	TSCL—Transmit single collision packet counter register	R/W	0x0000_0000	14.5.3.7.32/14-73
0x2_4700	TMCL—Transmit multiple collision packet counter register	R/W	0x0000_0000	14.5.3.7.33/14-74
0x2_4704	TLCL—Transmit late collision packet counter register	R/W	0x0000_0000	14.5.3.7.34/14-74
0x2_4708	TXCL—Transmit excessive collision packet counter register	R/W	0x0000_0000	14.5.3.7.35/14-75
0x2_470C	TNCL—Transmit total collision counter register	R/W	0x0000_0000	14.5.3.7.36/14-75
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter register	R/W	0x0000_0000	14.5.3.7.37/14-76
0x2_4718	TJBR—Transmit jabber frame counter register	R/W	0x0000_0000	14.5.3.7.38/14-76
0x2_471C	TFCS—Transmit FCS error counter register	R/W	0x0000_0000	14.5.3.7.39/14-77
0x2_4720	TXCF—Transmit control frame counter register	R/W	0x0000_0000	14.5.3.7.40/14-77
0x2_4724	TOVR—Transmit oversize frame counter register	R/W	0x0000_0000	14.5.3.7.41/14-78

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4728	TUND—Transmit undersize frame counter register	R/W	0x0000_0000	14.5.3.7.42/14-78
0x2_472C	TFRG—Transmit fragments frame counter register	R/W	0x0000_0000	14.5.3.7.43/14-79
TSEC1 General Registers				
0x2_4730	CAR1—Carry register one register	R/W	0x0000_0000	14.5.3.7.44/14-79
0x2_4734	CAR2—Carry register two register	R/W	0x0000_0000	14.5.3.7.45/14-80
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE01_FFFF	14.5.3.7.46/14-82
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFF	14.5.3.7.47/14-83
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—
TSEC1 Hash Function Registers				
0x2_4800	IADDR0—Individual address register 0	R/W	0x0000_0000	14.5.3.8.1/14-84
0x2_4804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_4808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_480C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_4810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_4814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_4818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_481C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_4820–0x2_487C	Reserved	R	0x0000_0000	—
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	14.5.3.8.2/14-85
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0–0x2_4AFF	Reserved	R	0x0000_0000	—
TSEC1 Attribute Registers				
0x2_4B00–0x2_4BF4	Reserved	R	0x0000_0000	—
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	14.5.3.9.1/14-85

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	14.5.3.9.2/14-87
TSEC1 Future Expansion Space				
0x2_4C00–0x2_4FFF	Reserved	R	0x0000_0000	—
TSEC2 Registers				
0x2_5000–0x2_5FFF	TSEC2 registers **Note:** TSEC2 has the same memory-mapped registers that are described for TSEC1 from 0x 2_4000 to 0x2_4FFF except the offsets are from 0x 2_5000 to 0x2_5FFF.			

[1] R = means read-only, W = write only, R/W = read and write, LH = latches high, SC = self-clearing.

14.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the TSEC registers. Because all of the TSEC registers are 32 bits wide, only 32-bit register accesses are supported.

14.5.3.1 TSEC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

14.5.3.1.1 Interrupt Event Register (IEVENT)

If an event occurs that sets a bit in the interrupt event (IEVENT) register, shown in Figure 14-6, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. Clearing the IEVENT bit clears the interrupt signal. The bit in the IEVENT register is cleared if a 1 is written to that bit position. A write of 0 has no effect.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts that may occur in normal operation are:

- GTSC, GRSC, TXF, TXB, TXC, RXF, RXB, RXC, MMRD, MMWR, and MSRO

Interrupts resulting from errors/problems detected in the network or transceiver are:

- BABR, BABT, LC, and CRL/XDA

Interrupts resulting from internal errors are:

- EBERR, XFUN, and BSY

Some of the error interrupts are independently counted in the management information base (MIB) block counters. Software may choose to mask off these interrupts because these errors are visible to network management through the MIB counters.

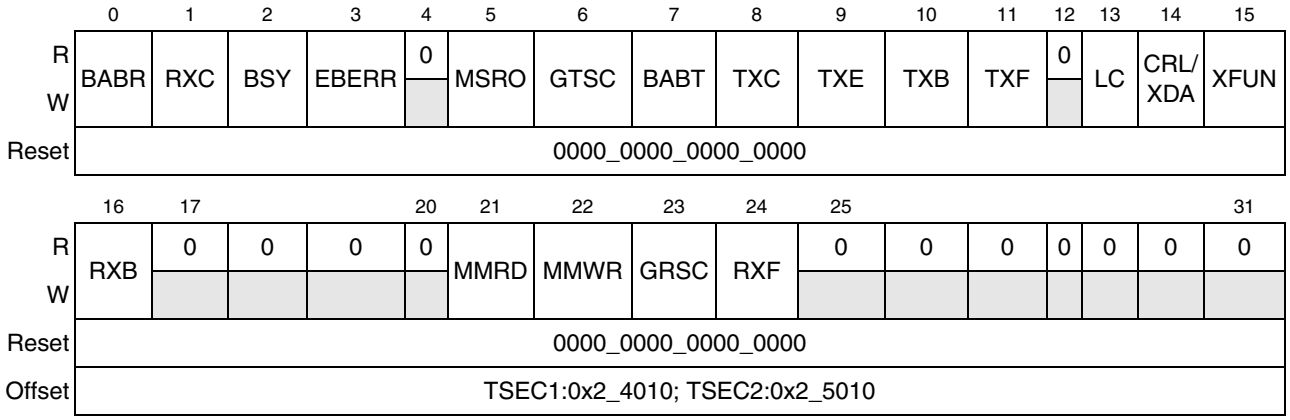


Figure 14-6. IEVENT Register Definition

Table 14-4 describes the fields of the IEVENT register.

Table 14-4. IEVENT Field Descriptions

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[Huge Frame] is set. 0 Excessive frame not received. 1 Excessive frame received.
1	RXC	Receive control interrupt. A control frame was received. If MACCFG1[Rx_Flow] is set, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was received. 0 Control frame not received. 1 Control frame received.
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. When IEVENT[BSY] is set RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit and RSTAT[QHLT] are set whenever the TSEC reads an RxBD with its Empty field cleared. 0 No frame received and discarded. 1 Frame received and discarded.
3	EBERR	Ethernet bus error. This bit indicates that a system bus error for a memory read occurred while a DMA transaction was underway. If the EBERR is set while transmission is in progress, the DMA stops sending data to the Tx FIFO which eventually causes an underrun error (XFUN) and TSTAT[THLT] is set. If the EBERR is set while receiving a frame, the DMA discards the frame and RSTAT[QHLT] is set. 0 No system bus error occurred. 1 System bus error occurred.
4	—	Reserved
5	MSRO	MSTAT Register Overflow. This interrupt is asserted if the count for one of the MSTAT registers has exceeded the size of the register. 0 MSTAT count not exceeding register size. 1 MSTAT count exceeds register size.

Table 14-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
6	GTSC	Graceful transmit stop complete. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. This interrupt is asserted for one of two reasons. • A graceful stop, which was initiated by setting DMACTRL[GTS], is now complete. • A graceful stop, which was initiated by setting TCTRL[TFC_PAUSE], is now complete. 0 Graceful transmit stop not complete or not initiated. 1 Graceful transmit stop completed.
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's Maximum Frame Length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame. 0 Transmitted frame length not exceeding maximum frame length. 1 Transmitted frame length exceeding maximum frame length.
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted. 1 Control frame transmitted.
9	TXE	Transmit error. This bit indicates that an error occurred on the transmitted channel that has caused TSTAT[THLT] to be set by TSEC. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL/XDA, XFUN). It is not set if DMACTRL[WOP] is set and TSEC runs out of TxBDs to process. In order to begin transmitting packets again, the user must clear TSTAT[THLT]. 0 No transmit channel error occurred. 1 Transmit channel error occurred.
10	TXB	Transmit buffer. This bit indicates that a transmit buffer descriptor was updated whose I (Interrupt) bit was set in its status word and was not the last buffer descriptor of the frame. 0 No transmit buffer descriptor updated. 1 Transmit buffer descriptor updated.
11	TXF	Transmit frame interrupt. This bit indicates that a frame was transmitted and that the last corresponding Transmit Buffer Descriptor (TxBD) was updated. This only occurs if the I (Interrupt) bit in the status word of the Buffer Descriptor is set. 0 No frame transmitted/TxBD not updated. 1 Frame transmitted/TxBD updated.
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred. 1 Late collision occurred.
14	CRL/ XDA	Collision retry limit/excessive defer abort. This bit indicates either one of two conditions occurred while attempting to transmit a frame: 1) the number of successive transmission collisions has exceeded the MAC's HAFDUP[Retransmission Maximum] count or 2) an excessive defer abort condition has occurred. An excessive defer abort condition occurs when the TSEC waits more than 3036 bytes while attempting to send a frame and HAFDUP[EXCESS_DEFER] is 0. The TxBD[DEF] or OSTBD[DEF] is also set. In either case the frame is discarded without being transmitted and the TSEC is halted (TSTAT[THLT] is set). The CRL or XDA condition can only occur while in half-duplex mode. 0 Successive transmission collisions do not exceed maximum and/or no excessive defer abort condition has occurred. 1 Successive transmission collisions exceed maximum or an excessive defer abort condition has occurred.

Table 14-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun. 1 Transmit FIFO underrun.
16	RXB	Receive buffer. These bits indicate that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated. 1 Receiver buffer descriptor updated.
17–20	—	Reserved
21	MMRD	MII management read completion 0 MII management read not issued or in process. 1 MII management read completed that was initiated by a user through the MII Scan or Read cycle command.
22	MMWR	MII management write completion 0 MII management write not issued or in process. 1 MII management write completed that was initiated by a user write to the MIIMCON register.
23	GRSC	Graceful receive stop complete. This interrupt is asserted if a graceful receive stop is completed. It allows the user to know if the system has completed the stop and it is safe to write to receive registers (status, control or configuration registers) that are used by the system during normal operation. 0 Graceful stop not completed. 1 Graceful stop completed.
24	RXF	Receive frame interrupt. The last receive buffer descriptor (RxBD) of a frame was updated. This occurs only if the I (interrupt) bit in the buffer descriptor status word is set. 0 Frame not received. 1 Frame received.
25–31	—	Reserved

14.5.3.1.2 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events are allowed to generate an actual interrupt. All implemented bits in this CSR are R/W. This register is cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, an interrupt is generated. The interrupt signal can be cleared by clearing the corresponding IEVENT bit.

Figure 14-7 shows the IMASK register.

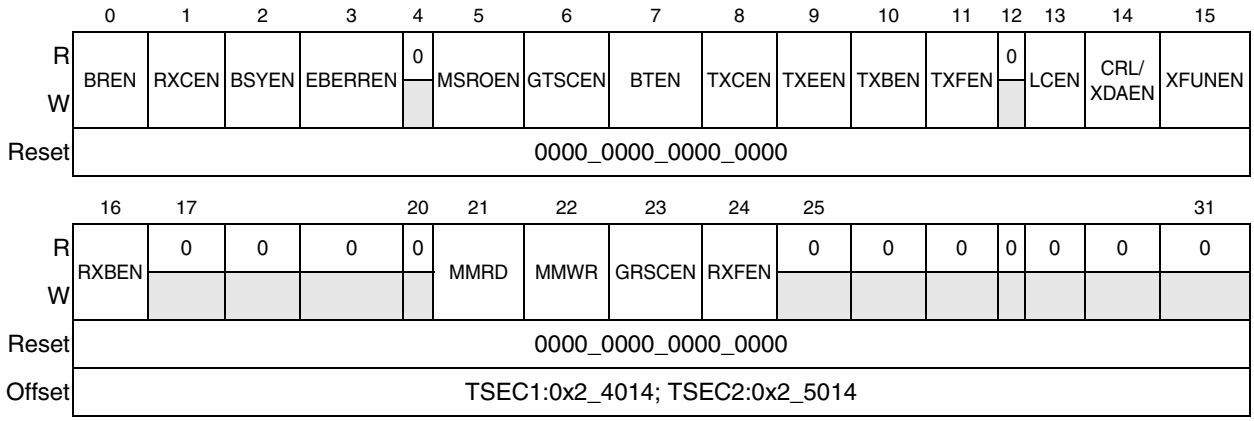


Figure 14-7. IMASK Register Definition

Table 14-5 describes the fields of the IMASK register.

Table 14-5. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MSTAT register overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRL/XDAEN	Collision retry limit/excessive defer enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–20	—	Reserved
21	MMRD	MII management read completion enable
22	MMWR	MII management write completion enable

Table 14-5. IMASK Field Descriptions (continued)

Bits	Name	Description
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–31	—	Reserved

14.5.3.1.3 Error Disabled Register (EDIS)

The error disabled register, shown in Figure 14-8, controls error reporting by the TSEC. The IEVENT bit corresponding to an error will not be set if the error is disabled in EDIS.

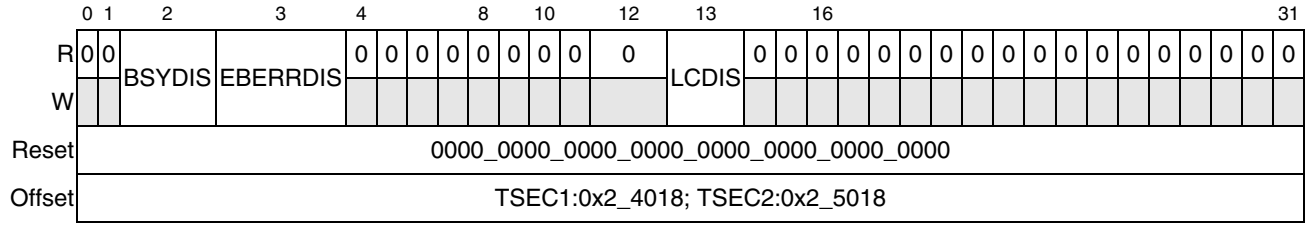


Figure 14-8. Error Disabled Register (EDIS)

Table 14-6 describes the fields of the EDIS register.

Table 14-6. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable 0 Allow TSEC to report IEVENT[BSY] status and halt buffer descriptor queue if BSY condition occurs. 1 Do not set IEVENT[BSY] and do not halt buffer descriptor queue if BSY condition occurs.
3	EBERRDIS	Ethernet controller bus error disable 0 Allow TSEC to report IEVENT[EBERR] status and halt buffer descriptor queue if EBERR condition occurs. 1 Do not set IEVENT[EBERR] and do not halt buffer descriptor queue if EBERR condition occurs.
4–12	—	Reserved
13	LCDIS	Late collision disable 0 Allow TSEC to report IEVENT[LC] status, set the buffer descriptor LC field, and halt buffer descriptor queue if LC condition occurs. 1 Do not set IEVENT[LC] nor the buffer descriptor LC field, and do not halt buffer descriptor queue if LC condition occurs.
14–31	—	Reserved

14.5.3.1.4 Ethernet Control Register (ECNTRL)

ECNTRL, shown in Figure 14-9, is used to reset, configure, and initialize the TSEC.

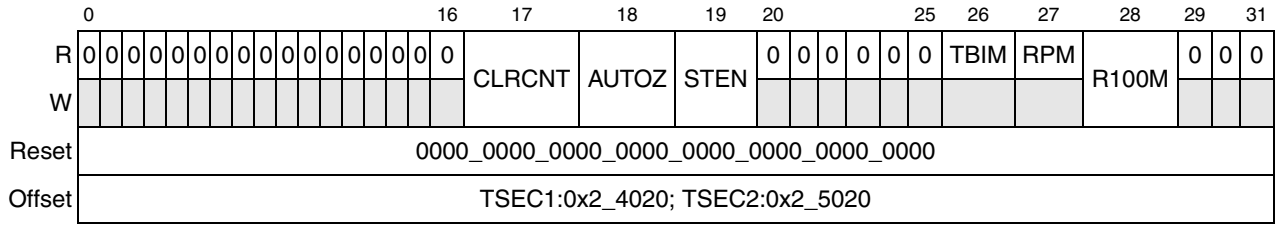


Figure 14-9. ECNTRL Register Definition

Table 14-7 describes the fields of the ECNTRL register.

Table 14-7. ECNTRL Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	CLRCNT	Clear all statistics counters 0 Allow MSTAT counters to continue to increment. 1 Reset all MSTAT counters. This bit is self-resetting.
18	AUTOZ	Automatically zero addressed statistical counter values, input to MSTAT module. 0 The user must write the addressed counter zero after a host read. 1 The addressed counter value is zeroed on a host read. This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
19	STEN	Statistics enabled, input to MSTAT module. 0 Statistics not enabled 1 Enables internal counters to update This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
20–25	—	Reserved
26	TBIM	Ten-bit interface mode. If this bit is set, ten-bit interface mode is enabled. This bit can be signal configured at reset to set or clear using the TSEC*n*_GTX_CLK signal. 0 GMII or MII mode interface 1 TBI mode interface
27	RPM	Reduced signal mode. If this bit is set, a reduced signal interface is expected. ECNTRL[TBI Mode] selects GMII or TBI. If RPM is selected, the MAC must be in GMII or TBI mode (RMII is not supported). This register can be signal configured at reset to 0 or 1 with the EC_MDC signal. 0 GMII or TBI in non-reduced signal mode configuration 1 RGMII or RTBI reduced signal mode
28	R100M	RGMII 100 mode. This bit is ignored unless RPM is set and MACCFG2[I/F Mode] is assigned to 10/100 (01). If this bit is set, the TSEC interface is in 100 Mbps speed. 0 RGMII is in 10 Mbps mode 1 RGMII is in 100 Mbps mode
29–31	—	Reserved

14.5.3.1.5 Minimum Frame Length Register (MINFLR)

MINFLR is written by the user. It tells TSEC the smallest packet to accept and place in a receive buffer pointed to by the RxBD. Figure 14-10 shows the MINFLR register.

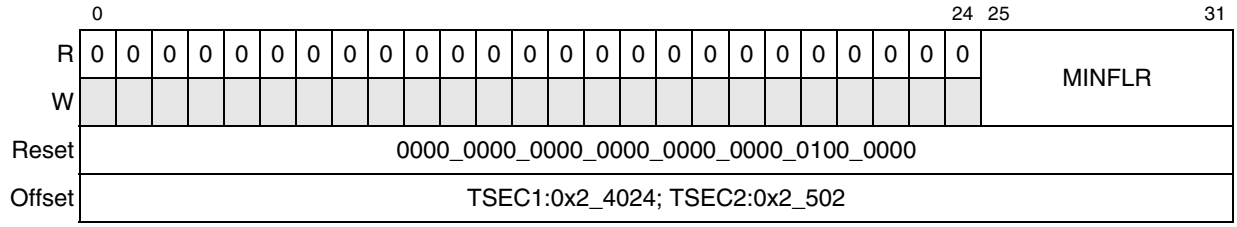


Figure 14-10. MINFLR Register Definition

Table 14-8 describes the fields of the MINFLR register.

Table 14-8. MINFLR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25–31	MINFLR	Minimum receive frame length (typically 64 decimal). If the Ethernet receives an incoming frame shorter than MINFLR, it discards that frame unless RCTRL[RSF] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The largest allowable value for MINFLR is 64. Unlike the MPC8260, in which PADs are added to make the transmit frame equal to MINFLR bytes, if padding is requested, TSEC always PADS transmit frames to 64 bytes ignoring MINFLR. MINFLR is only used to determine the minimum size of acceptable receive frames.

14.5.3.1.6 Pause Time Value Register (PTV)

PTV, shown in Figure 14-11, is written by the user to store the pause duration used when the TSEC initiates a pause frame through TCTRL[TFC_PAUSE]. The low-order 16 bits (PT) represent the pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of pause_quanta, equal to 512 bit times. The pause time can range from 0 to 65,535 pause_quanta, or 0 to 33,553,920 bit times. See Section 14.6.2.7, “Flow Control,” for additional details.

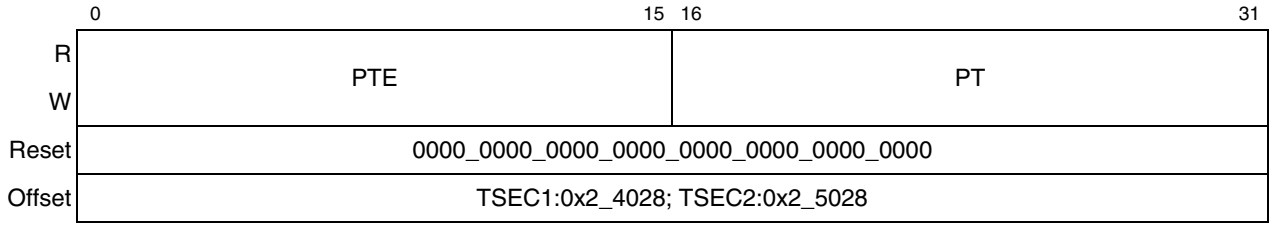


Figure 14-11. PTV Register Definition

Table 14-9 describes the fields of the PTV register.

Table 14-9. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the pause frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 pause frame format requires this parameter to be set to 0.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (that is, 512 bit times). This pause value is used as part of the pause frame to be sent when TCTRL[TFC_PAUSE] is set. See Section 14.6.2.7, “Flow Control,” for more information.

14.5.3.1.7 DMA Control Register (DMACTRL)

DMACTRL, shown in Figure 14-12, is used to configure the DMA block. register.

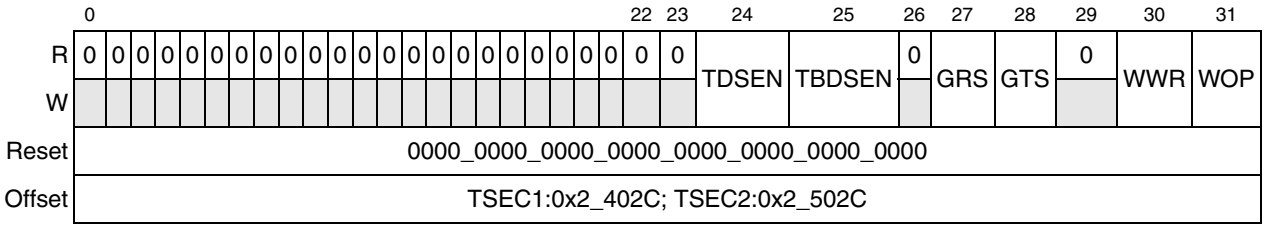


Figure 14-12. DMACTRL Register Definition

Table 14-10 describes the fields of the DMACTRL register.

Table 14-10. DMACTRL Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TDSEN	Tx Data snoop enable 0 Disables snooping of all transmit frames from memory. 1 Enables snooping of all transmit frames from memory.
25	TBDSEN	TxBD snoop enable 0 Disables snooping of all transmit BD memory accesses. 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved
27	GRS	Graceful receive stop. If this bit is set, the Ethernet controller stops receiving frames following completion of the frame currently being received. (That is, after a valid end of frame was received). The buffer of the receive frame associated with the EOF is closed and the IEVENT[GRSC] is set. Because the MAC’s receive enable bit (MACCFG[Rx_EN]) may still be set, the MAC may continue to receive but the TSEC ignores the receive data until GRS is cleared. If this bit is cleared, the TSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxBD. If GRS is set, the user must monitor the graceful receive stop complete (GRSC) bit in the IEVENT register to insure that the graceful receive stop was completed. The user can then clear IEVENT[GRSC] and can write to receive registers that are accessible to both user and the TSEC hardware without fear of conflict. 0 TSEC scans input data stream for valid frame. 1 TSEC stops receiving frames following completion of current frame.

Table 14-10. DMACTRL Field Descriptions (continued)

Bits	Name	Description
28	GTS	Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after the completion of any frame that is currently being transmitted and the GTSC interrupt in the IEVENT register is asserted. If frame transmission is not currently underway, the GTSC interrupt is asserted immediately. Once transmission has completed, a “restart” can be accomplished by clearing GTS. 0 Controller continues. 1 Controller stops transmission after completion of current frame.
29	—	Reserved
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, IE, XFUN, LC, CRL/XDA, RXB, RXF, the TSEC waits for acknowledgement from system that the transmit or receive BD being updated was stored in memory.
31	WOP	Wait or poll. This bit, which is applicable only to the transmitter, provides the user the option for the TSEC to periodically poll TxBD or to wait for software to tell TSEC to fetch a buffer descriptor. While operating in the “Wait” mode, the TSEC allows two additional reads of a descriptor which is not ready before entering a halt state. No interrupt is driven. (IEVENT[TXE] is clear.) To resume transmission, software must clear TSTAT[THLT]. Note that if this bit is set, the user must ensure that all TxBDs involved with sending a frame have their ready bits set before any transmission begins. Otherwise, TSEC behaves in a boundedly undefined fashion. A buffer descriptor is considered not ready when its TxBD[Ready] field is clear or its TxBD[Data Length] field is zero. It is considered ready when both TxBD[Ready] is set and TxBD[Data Length] is non-zero. In Wait mode (DMACTRL[WOP] is set) when TSEC is processing a frame and an intermediate TxBD’s ready bit is cleared, TSEC does not halt, but instead continuously polls the same TxBD until the TxBD becomes ready or an Ethernet interface error or a memory error is encountered. If the TxBD becomes ready, TSEC continues to process the frame. If an error occurs before the TxBD becomes ready, TSEC reports the error in both the IEVENT register as well as the TxBD for Ethernet interface errors, or the IEVENT[EBERR] for a memory error and sets the TSTAT[THLT] bit. Note that software must eventually set all of its TxBDs for a frame, because TSEC continuously reads an intermediate TxBD until it becomes ready if insufficient data has been read to surpass the FIFO Transmit Threshold (FIFO_TX_THR) register value. Note, the polling in Wait mode differs from the polling in Poll mode. In Poll mode polling is performed every 512 Ethernet bus Tx clocks. In Wait mode polling is performed as fast as possible after the TxBD[Ready] = 0 is read. If software is slow in setting all TxBD Ready bits of a frame to a one, then the system performance may decrease. 0 Poll TxBD every 512 clocks. 1 Do not poll, but wait for a write to TSTAT[THLT].

14.5.3.1.8 TBI Physical Address Register (TBIPA)

This TBIPA, shown in Figure 14-13, is writable by the user to assign a physical address to the TBI for MII management configuration. The TBI registers are accessed at the offset of TBIPA. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to Section 14.5.4, “Ten-Bit Interface (TBI).”

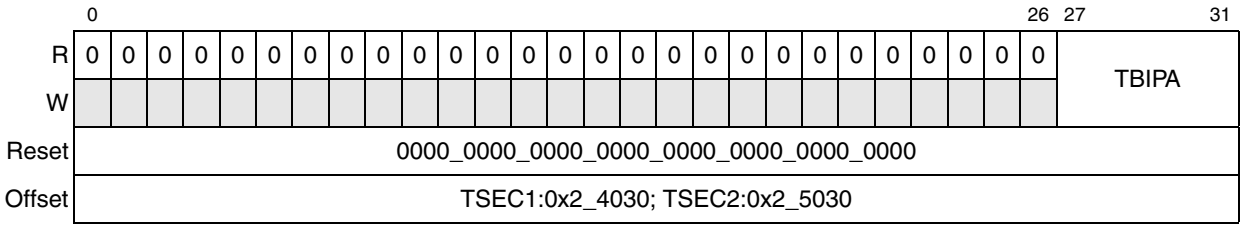


Figure 14-13. TBIPA Register Definition

Table 14-11 describes the fields of the TBIPA register.

Table 14-11. TBIPA Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	TBIPA	TBI PHY address. This field is used to program the PHY address of the ten-bit interface's MII management bus. To access the TBI register the user must write the TBIPA value to the MIIMADD[PHY Address] register located in the MAC register section. Refer to Section 14.5.3.6.8, "MII Management Address Register (MIIMADD)."

14.5.3.2 TSEC FIFO Control and Status Registers

The following registers allow the user to change some of the default settings in the FIFO that can be used to optimize operation for performance or for safety. They must be set carefully in order to avoid an underrun condition. Underrun is an error condition in which data is not retrieved from external memory quickly enough, leaving the TX FIFO empty before the complete frame is transmitted. Because different combinations of events, several of which are determined by the user, can lead to underrun, the TSEC provides several FIFO registers that allow the user to select the proper setting to be able to tune the system and obtain the maximum performance with minimal chance of underrun. The principal causes for underrun in the TSEC are:

- Misaligned data buffer addresses
- Small data buffer sizes
- Combinations of the above

It is recommended that the minimum size data buffers be 64 bytes and that data buffers be 64-byte aligned. The user can deviate from these recommended values to try to increase performance or to use less memory, but unless the default values of some of the FIFO registers are adjusted, the probability of an underrun may also increase. The FIFO_TX_THR (default is 256 entries or 1 Kbyte) indicates the amount of data required to be in the FIFO before starting the transmission of a frame. The FIFO_TX_STARVE (default is 128 entries or 512 bytes) is used to indicate that the amount of data in the FIFO is so low that the risk of underrun is extremely high. The FIFO_TX_STARVE_SHUTOFF (default is 256 entries or 1 Kbyte) contains the watermark level to be used for exiting the starve state. These registers are intended to allow the user to make the proper trade-off. If triggered, the starve mode, for instance, automatically raises the priority of TSEC fetches from memory.

14.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 14-14, is writable by the user to configure the properties of the TSEC FIFO.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16													29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TFC_PAUSE_EN	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_404C; TSEC2:0x2_504C															

Figure 14-14. FIFO_PAUSE_CTRL Register Definition

Table 14-12 describes the fields of the FIFO_PAUSE_CTRL register.

Table 14-12. FIFO_PAUSE_CTRL Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	TFC_PAUSE_EN	TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset but should always be set during initialization as undefined behavior results during normal operation when left cleared. 0 Pause control frame transmission disabled (default, but must be set during initialization). 1 Pause control frame transmission enabled.
31	—	Reserved

14.5.3.2.2 FIFO Transmit Threshold Register (FIFO_TX_THR)

The main purpose of the threshold register is to trigger the unloading of FIFO data to the PHY. It represents the numerical SRAM entry (0–511 for 2-Kbyte FIFO) to trigger the threshold function. If the number of valid entries in the FIFO is equal to or greater than the threshold register, transmission can begin. This register is read/write by software and is initialized to 0000_0000_0000_0000_0000_0001_0000_0000 at system reset. Figure 14-15 shows the FIFO_TX_THR register.

	0																					22	23	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FIFO_TX_THR	
W																								
Reset	0000_0000_0000_0000_0000_0001_0000_0000																							
Offset	TSEC1:0x2_408C; TSEC2:0x2_508C																							

Figure 14-15. FIFO_TX_THR Register Definition

Table 14-13 describes the fields of the FIFO_TX_THR register.

Table 14-13. FIFO_TX_THR Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_THR	FIFO transmit threshold. These bits mark the number of entries in the transmit FIFO that, if reached, trigger the unloading of frame data into the MAC.

14.5.3.2.3 FIFO Transmit Starve Register (FIFO_TX_STARVE)

The purpose of the starve register, shown in Figure 14-16, is to inform the system of extremely imminent underrun conditions. It represents the numerical SRAM entry (0-511 for 2-Kbyte FIFO) to trigger the starve function. If the number of valid entries in the FIFO is less than or equal to the starve register, a starve alert is triggered.

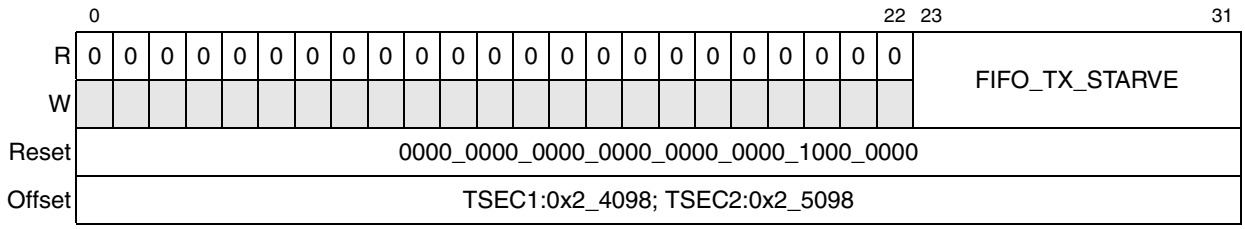


Figure 14-16. FIFO_TX_STARVE Register Definition

Table 14-14 describes the fields of the FIFO_TX_STARVE register.

Table 14-14. FIFO_TX_STARVE Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_STARVE	FIFO transmit starve. These bits indicate the value to trigger the transmit starve function. It triggers once the number of valid entries in the FIFO is less than or equal to the FIFO Tx starve. The starve state turns off if the number of valid entries in the FIFO becomes greater than or equal to the FIFO Tx starve shutoff register.

14.5.3.2.4 FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)

The starve shutoff register, shown in Figure 14-17, contains the watermark level to be used for exiting the starve state. If the starve state is in effect and the number of valid entries in the FIFO becomes greater than or equal to the value in the FIFO transmit starve shutoff register, the starve condition ends. This register is read/write by software.

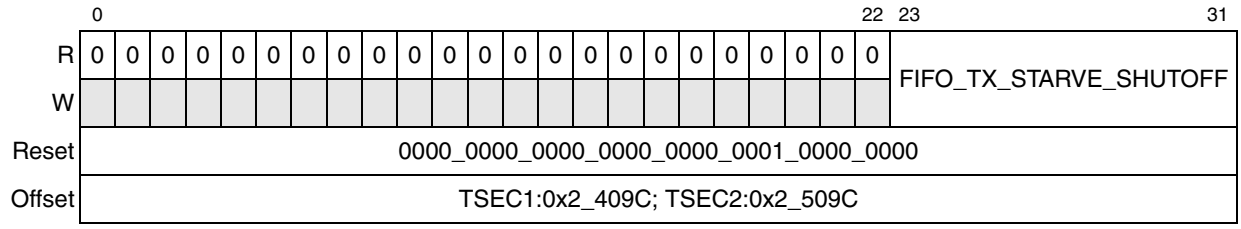


Figure 14-17. FIFO_TX_STARVE_SHUTOFF Register Definition

Table 14-15 describes the fields of the FIFO transmit starve shutoff register.

Table 14-15. FIFO_TX_STARVE_SHUTOFF Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_STARVE_SHUTOFF	FIFO transmit starve shutoff. Indicates the value beyond which to exit the starve state. The starve state turns off if the number of valid entries in the FIFO becomes greater than or equal to the FIFO Tx starve shutoff register.

14.5.3.3 TSEC Transmit Control and Status Registers

This section describes the control and status registers that are used specifically for transmitting Ethernet frames. All of the registers are 32 bits wide.

14.5.3.3.1 Transmit Control Register (TCTRL)

TCTRL, shown in Figure 14-18, is writable by the user to configure the transmit block.

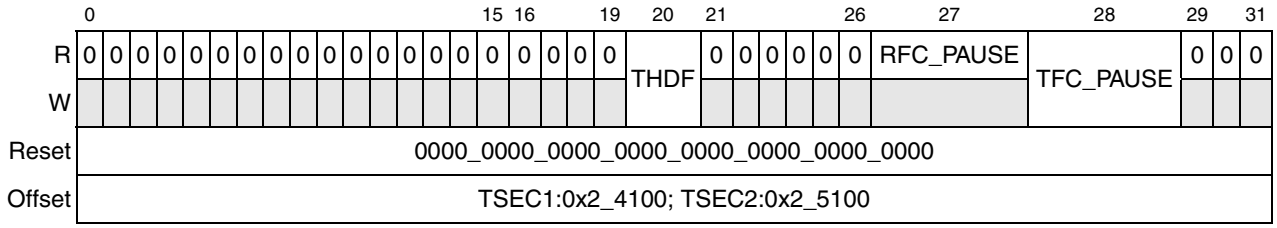


Figure 14-18. TCTRL Register Definition

Table 14-16 describes the fields of the TCTRL register.

Table 14-16. TCTRL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20	THDF	Transmit half-duplex flow control. Written by user. This bit is not self-resetting. 0 Disable back pressure. 1 Back pressure is applied to media.
21–26	—	Reserved

Table 14-16. TCTRL Field Descriptions (continued)

Bits	Name	Description
27	RFC_PAUSE	Receive flow control pause frame. Written by TSEC. This read-only status bit is set if a flow control pause frame was received and the transmitter is paused for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. 0 Pause duration complete. 1 Pause duration in progress.
28	TFC_PAUSE	Transmit flow control pause frame. Use this bit to transmit a PAUSE frame. To transmit a flow control pause frame, first set FIFO_PAUSE_CTRL[TFC_PAUSE_EN]. Next, set MACCFG1[GTS]. If TFC_PAUSE is then set, the MAC stops transmission of data frames after the current transmission completes. The GTSC interrupt in the IEVENT register is asserted. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame with the duration value obtained from the PTV register. The TXC interrupt occurs after sending the control pause frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. Note that if the transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, the MAC may still transmit a MAC control PAUSE frame. 0 No outstanding pause frame transmission request. 1 Pause frame transmission requested.
29–31	—	Reserved

14.5.3.3.2 Transmit Status Register (TSTAT)

TSTAT, shown in Figure 14-19, is written by TSEC to convey DMA status information.

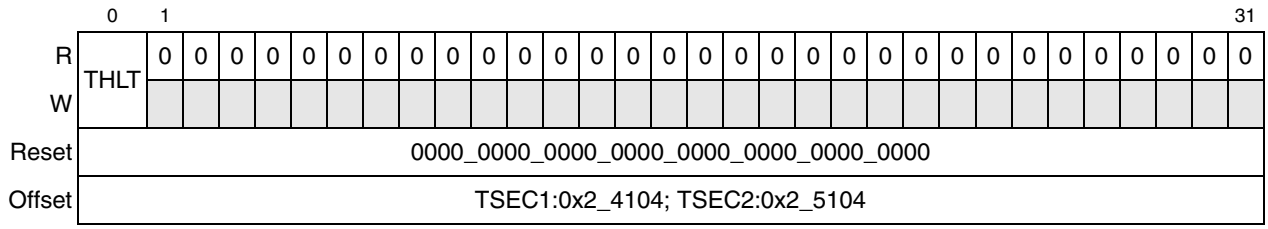


Figure 14-19. TSTAT Register Definition

Table 14-17 describes the fields of the TSTAT register.

Table 14-17. TSTAT Field Descriptions

Bits	Name	Description
0	THLT	Transmit halt 0 No hardware-initiated transmission halt. 1 TSEC transmission function halted. This bit is written by TSEC to inform the user that it is no longer processing transmit frames and that the transmit DMA function is disabled by hardware. To restart the transmission function, the user must clear this bit by writing a one.
1–31	—	Reserved

14.5.3.3.3 TxBD Data Length Register (TBDLEN)

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit buffer. Figure 14-20 shows the TBDLEN register.

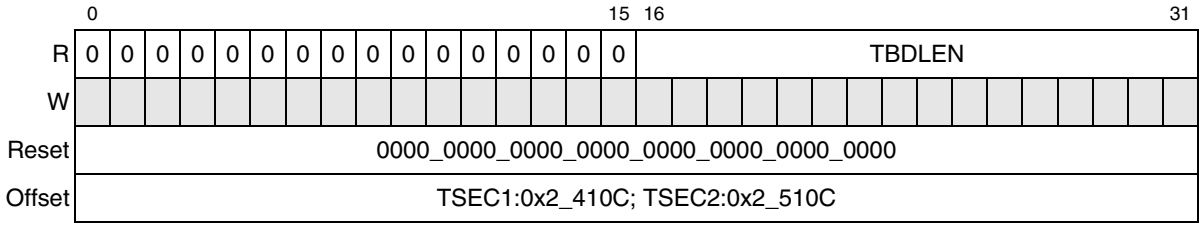


Figure 14-20. TBDLEN Register Definition

Table 14-18 describes the fields of the TBDLEN register.

Table 14-18. TBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TBDLEN	Internally written by the DMA module. The transmit channel remains active until TBDLEN is 0.

14.5.3.3.4 Transmit Interrupt Coalescing Configuration Register (TXIC)

TXIC, shown in Figure 14-21, enables and configures the operational parameters for interrupt coalescing associated with transmitted frames. Refer to Section 14.6.2.8.1, "Interrupt Coalescing," for a functional description of interrupt coalescing.

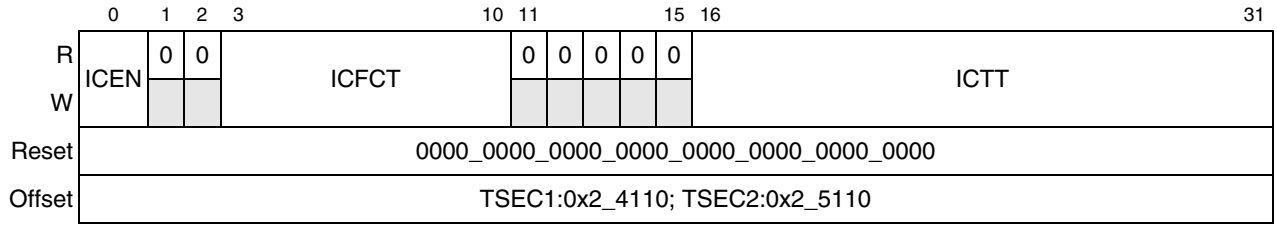


Figure 14-21. TXIC Register Definition

Table 14-18 describes the fields of the TXIC register.

Table 14-19. TXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received if the TSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set). 1 Interrupt coalescing is enabled. If the TSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by TXIC[ICFCT]) or when the threshold timer expires (defined by TXIC[ICTT]).
1–2	—	Reserved

Table 14-19. TXIC Field Descriptions (continued)

Bits	Name	Description
3–10	ICFCT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines how many frames are transmitted before raising an interrupt[1]. Valid values for this field are from 1 to 255. A value of 0 is illegal. If set to 0, an interrupt can only be cleared by first clearing TXIC[ICEN] and then clearing the IEVENT[TXF] bit. Note that a value of 1 functionally defeats the advantages of interrupt coalescing since the frame threshold is reached with each frame transmitted.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines the maximum amount of time after transmitting a frame before raising an interrupt[1], subject also to IMASK[TXFEN]. If frames have been transmitted but the frame count threshold has not been met, an interrupt is raised when the threshold timer expires. The threshold timer is reset once an interrupt has been asserted. It begins counting once the interrupt is cleared and IEVENT[TXF] is set. The threshold value is represented in units equal to 64 TSEC interface clocks. Valid values for this field are from 1 to 65535. A value of 0 is illegal and results in behavior identical to that when interrupt coalescing is disabled (TXIC[ICEN] is cleared).

[1] Interrupts resulting from the Interrupt bit (I) of the buffer descriptor in question and enabled subject to the IMASK register (IMASK[TXFEN] is set).

14.5.3.3.5 Current Transmit Buffer Descriptor Pointer Register (CTBPTR)

CTBPTR contains the address of the transmit buffer descriptor either currently being processed, or processed most recently. Figure 14-22 shows the CTBPTR register.

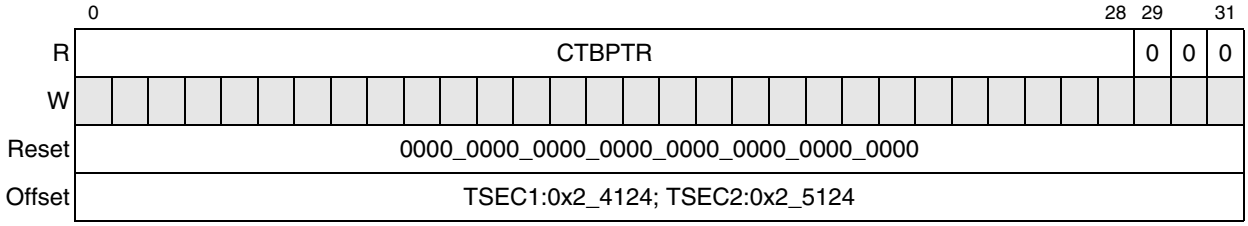


Figure 14-22. CTBPTR Register Definition

Table 14-20 describes the fields of the CTBPTR register.

Table 14-20. CTBPTR Field Descriptions

Bits	Name	Description
0–28	CTBPTR	The CTBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.3.6 Transmit Buffer Descriptor Pointer Register (TBPTR)

TBPTR, shown in Figure 14-23, contains the low-order 32 bits of the next transmit buffer descriptor address. TBPTR takes on the TBASE value when TBASE is written by software. Although not necessary

in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by IEVENT[GTSC] or TSTAT[THLT].

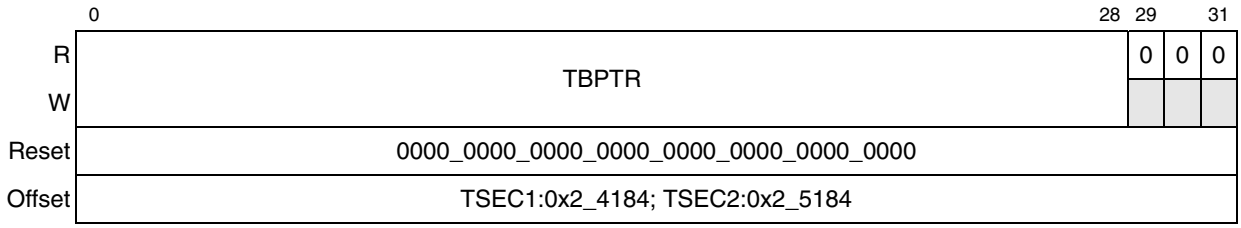


Figure 14-23. TBPTR Register Definition

Table 14-21 describes the fields of the TBPTR register.

Table 14-21. TBPTR Field Descriptions

Bits	Name	Description
0–28	TBPTR	The TBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.3.7 Transmit Descriptor Base Address Register (TBASE)

The TBASE register, shown in Figure 14-24, is written by the user with the TxBD base address. The value must be divisible by eight for the 8-byte data buffer descriptors.

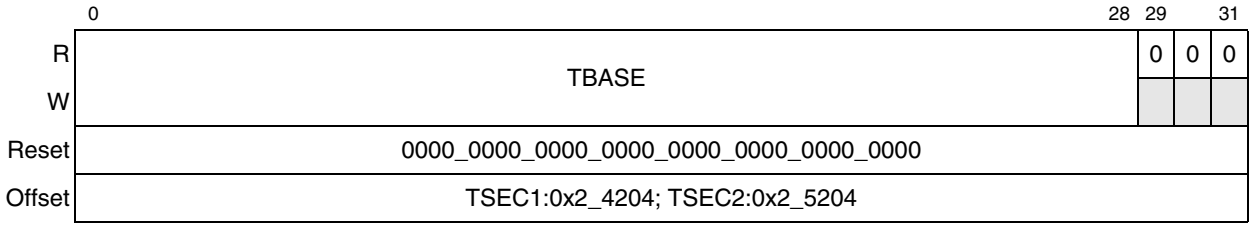


Figure 14-24. TBASE Register Definition

Table 14-22 describes the fields of the TBASE register.

Table 14-22. TBASE Field Descriptions

Bits	Name	Description
0–28	TBASE	Transmit base. TBASE defines the starting location in the memory map for the TSEC TxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the transmit packets. The user must initialize TBASE before enabling the TSEC transmit function.
29–31	—	Reserved

14.5.3.3.8 Out-of-Sequence TxBD Register (OSTBD)

The out-of-sequence TxBD register, OSTBD, shown in Figure 14-25, includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. If a flow control frame is sent

and OSTBD[I] is set, a TXC event is generated after frame transmission. This area must be cleared while not in use. Once the TSEC is in paused mode the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC regards it as a regular TxBD.

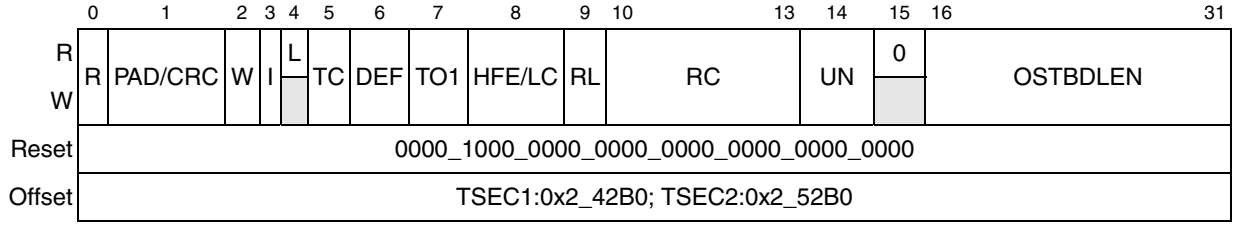


Figure 14-25. OSTBD Register Definition

Table 14-23 describes the fields of the OSTBD register.

Table 14-23. OSTBD Field Descriptions

Bits	Name	Description
0	R	Ready. Written by TSEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The TSEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which was prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
1	PAD/CRC	Padding and CRC attachment for short frames. (Valid only when MACCFG2[PAD/CRC] is cleared, and MACCFG2[CRC EN] bit is cleared.) If MACCFG2[PAD/CRC] is set, pads are added to all short frames; however, this bit is ignored. 0 Do not add PADs to short frames unless OSTBD[TC] is set. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADs up to MINFLR value, TSEC always PADs up to the IEEE minimum frame length of 64 bytes.
2	W	Wrap. Written by user. This bit is ignored by TSEC.
3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXF] is set after this buffer is serviced. This bit can cause an interrupt if IMASK[TXFEN] is enabled.
4	L	Last in frame. The OSTBD is always the last in the frame, so L is always set. (Hardwired to a value of 1.)
5	TC	Tx CRC. Written by user. (Valid only while it is set in the first BD and OSTBD[PAD/CRC] is cleared, MACCFG2[PAD/CRC] is cleared, and MACCFG2[CRC EN] is cleared.) If MACCFG2[PAD/CRC] is set or MACCFG2[CRC EN] is set, a CRC is added to all frames and this bit is ignored. 0 End transmission immediately after the last data byte, unless OSTBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. Written by TSEC. Hardware updates this bit after transmitting a frame if used as a ‘Defer indicator.’ Software/user updates this bit while building a transmit buffer descriptor if used as a ‘Hardware Event indicator.’ 0 This frame was not deferred. 1 This frame did not have a collision before it was sent but it was sent late because of deferring.
7	TO1	Transmit software ownership. This read/write bit may be utilized by software, as necessary. Its state does not affect the hardware nor is it affected by the hardware.

Table 14-23. OSTBD Field Descriptions (continued)

Bits	Name	Description
8	HFE/LC	Huge frame enable (written by user)/Late collision (written by TSEC) Huge frame enable. Written by user. Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's Maximum Frame Length register. 1 Do not truncate the transmit frame. Late collision. Written by TSEC. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The TSEC terminates the transmission and updates LC.
9	RL	Retransmission limit. Written by TSEC. 0 Transmission before maximum retry limit is hit. 1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The TSEC terminates the transmission and updates RL.
10–13	RC	Retry count. Written by TSEC. 0 The frame is sent correctly the first time. 1 More than zero attempts were needed to send the transmit frame. If this field is 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. Written by TSEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The TSEC terminates the transmission and updates UN.
15	—	Reserved
16–31	OSTBDLEN	Out-of-sequence TxBD data length. Written by user. Data length is the number of octets the TSEC transmits from this BD's data buffer. It is never modified by the TSEC. This field must be greater than zero in order for a transfer to take place.

14.5.3.3.9 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

The out-of-sequence Tx data buffer pointer register (OSTBDP), shown in Figure 14-26, contains the data buffer pointer fields in the same format as a regular TxBD, With OSTBD, it provides the complete 8-byte descriptor. This area must be cleared while not in use.

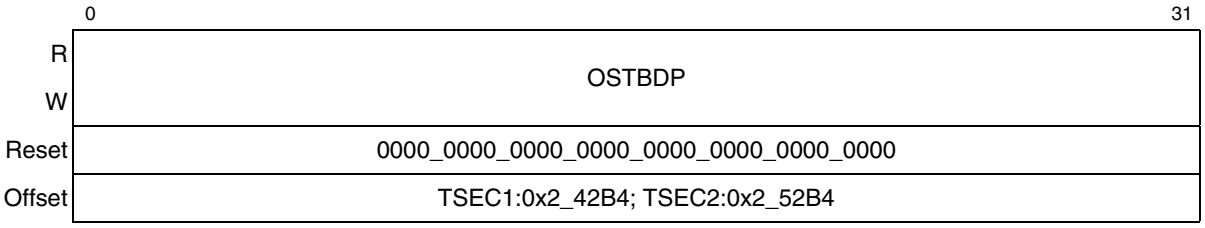


Figure 14-26. OSTBDP Register Definition

Table 14-24 describes OSTBDP.

Table 14-24. OSTBDP Field Descriptions

Bits	Name	Description
0–31	OSTDBP	Out-of-sequence Tx Data Buffer Pointer. Written by user. The transmit data buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

14.5.3.4 TSEC Receive Control and Status Registers

This section describes the control and status registers that are used specifically for receiving Ethernet frames. All of the registers are 32 bits wide.

14.5.3.4.1 Receive Control Register (RCTRL)

RCTRL, shown in Figure 14-27, controls the operational mode of the receive block. It must be written only after a system reset (at initialization) or if DMACTRL[GRS] is cleared.

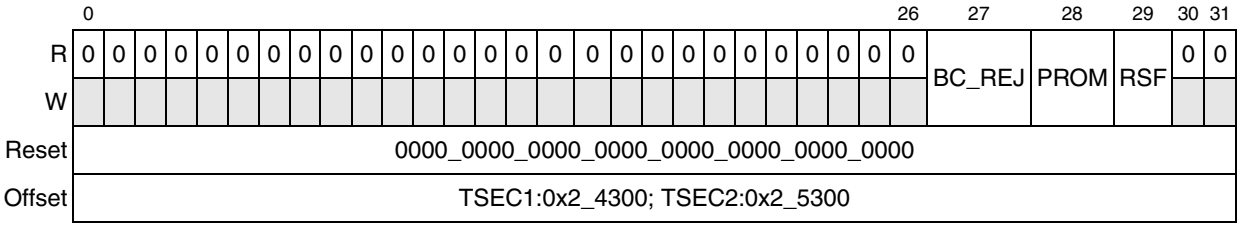


Figure 14-27. RCTRL Register Definition

Table 14-25 describes the fields of the RCTRL register.

Table 14-25. RCTRL Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD. 0 Broadcast frame reject is disabled. 1 Broadcast frame reject is enabled.
28	PROM	Promiscuous mode. When set, all frames except pause frames are accepted. 0 Promiscuous mode is disabled. 1 Promiscuous mode is enabled.
29	RSF	Receive short frame mode. When set, enables the reception of frames shorter than MINFLR bytes. Note that in order for short frames to be received when RSF is set, a DA hit must occur. When RSF is cleared, all frames shorter than MINFLR are automatically rejected. 0 Receive short frame mode is disabled. 1 Receive short frame mode is enabled.
30–31	—	Reserved

14.5.3.4.2 Receive Status Register (RSTAT)

TSEC writes to RSTAT, shown in Figure 14-28, under the following conditions:

- The receiver runs out of descriptors
- The receiver was halted because an error condition was encountered while receiving a frame

Software must clear the QHLT bit to take the TSEC receiver function out of a halt state.

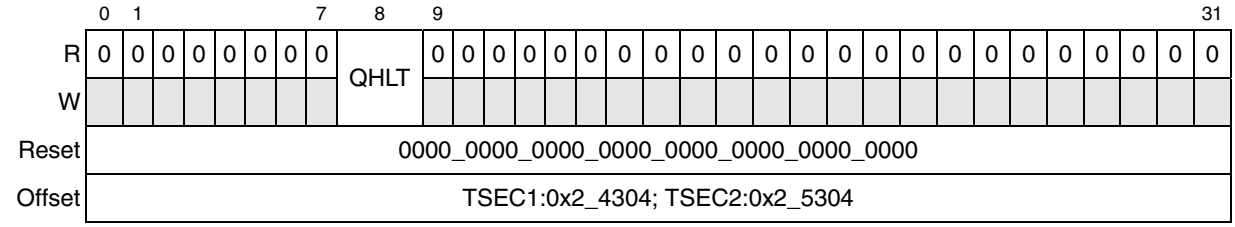


Figure 14-28. RSTAT Register Definition

Table 14-26 describes the fields of the RSTAT register.

Table 14-26. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT	RxBD queue is halted. When IEVENT[BSY] or IEVENT[EBERR] is set during reception of a packet, RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit is set whenever the TSEC reads an RxBD with its Empty field cleared or encounters a system bus error while processing an RX packet. It is a hardware-initiated stop indication (DMA_CTRL[GRS] being set by the user does not cause this bit to be set.). The current frame and all other frames directed to the halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.) 1 All Ethernet controller receive activity to RxBD queue is halted.
9–31	—	Reserved

14.5.3.4.3 RxBD Data Length Register (RBDLEN)

RBDLEN is a DMA register that contains the number of bytes remaining in the current receive buffer. Figure 14-29 shows the RBDLEN register.

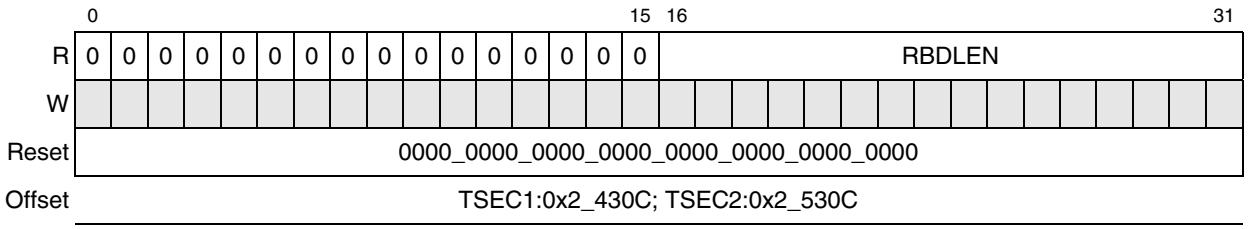


Figure 14-29. RBDLEN Register Definition

Table 14-27 describes the fields of the RBDLEN register.

Table 14-27. RBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RBDLEN	The RBDLEN is internally written by the DMA module. If RBDLEN is cleared, all activity in the receive channel stops.

14.5.3.4.4 Receive Interrupt Coalescing Configuration Register (RXIC)

The RXIC register enables and configures the operational parameters for interrupt coalescing associated with received frames. Refer to Section 14.6.2.8.1, "Interrupt Coalescing," for a functional description of interrupt coalescing as additional details regarding the use of this register. Figure 14-30 shows the RXIC register.

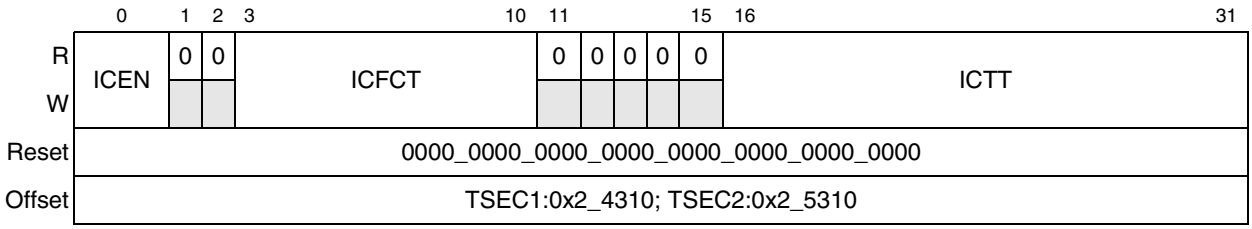


Figure 14-30. RXIC Register Definition

Table 14-18 describes the fields of the RXIC register.

Table 14-28. RXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received if the TSEC receive frame interrupt is enabled (IMASK[RXFEN] is set). 1 Interrupt coalescing is enabled. If the TSEC receive frame interrupt is enabled (IMASK[RXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by RXIC[ICFCT]) or when the threshold timer expires (defined by RXIC[ICTT]).
1–2	—	Reserved
3–10	ICFCT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (RXIC[ICEN] is set), this value determines how many frames are received before raising an interrupt[1]. Valid values for this field are from 1 to 255. A value of 0 is illegal. If set to 0, an interrupt can only be cleared by first clearing RXIC[ICEN] and then clearing the IEVENT[RXF] bit. Note that a value of 1 functionally defeats the advantages of interrupt coalescing since the frame threshold is reached with each frame received.

Table 14-28. RXIC Field Descriptions (continued)

Bits	Name	Description
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (RXIC[ICEN] is set), this value determines the maximum amount of time after receiving a frame before raising an interrupt[1], subject also to IMASK[RXFEN]. If frames have been received but the frame count threshold has not been met, an interrupt is raised when the threshold timer expires. The threshold timer is reset once an interrupt has been asserted. It begins counting once the interrupt is cleared and IEVENT[RXF] is set. The threshold value is represented in units equal to 64 TSEC interface clocks. Valid values for this field are from 1 to 65535. A value of 0 is illegal and results in behavior identical to that when interrupt coalescing is disabled (RXIC[ICEN] is cleared).

[1] Interrupts resulting from the Interrupt bit (I) of the buffer descriptor in question and enabled subject to the IMASK register (IMASK[RXFEN] is set).

14.5.3.4.5 Current Receive Buffer Descriptor Pointer Register (CRBPTR)

CRBPTR contains the address of the receive buffer descriptor either currently being processed, or processed most recently. Figure 14-31 shows the CRBPTR register.

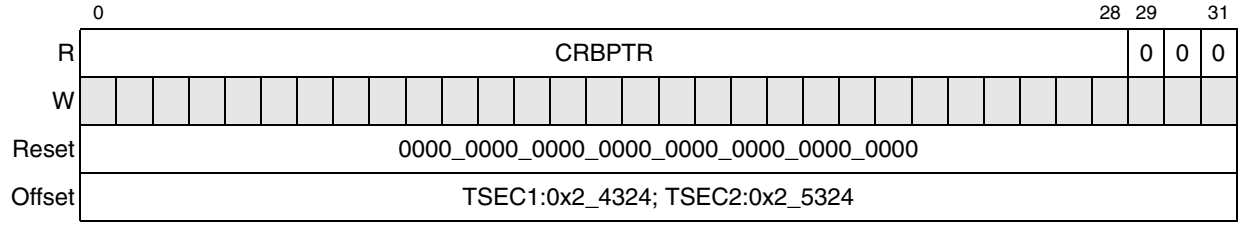


Figure 14-31. CRBPTR Register Definition

Table 14-29 describes the fields of the CRBPTR register.

Table 14-29. CRBPTR Field Descriptions

Bits	Name	Description
0–28	CRBPTR	The CRBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.4.6 Maximum Receive Buffer Length Register (MRBLR)

The MRBL register is written by the user. It informs the TSEC how much space is in the receive buffer pointed to by the RxBD. Figure 14-32 shows the MRBLR.

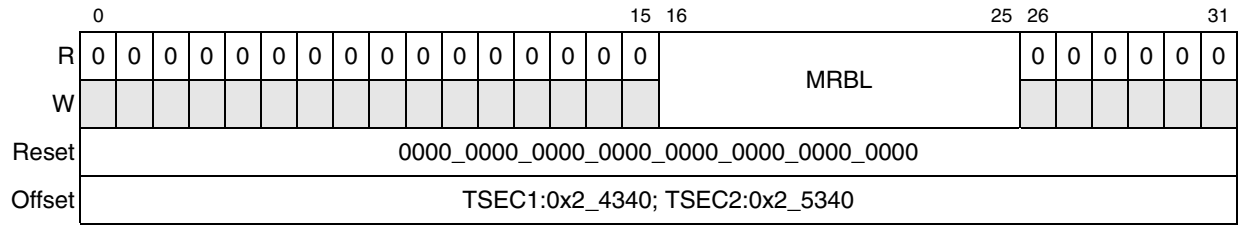


Figure 14-32. MRBL Register Definition

Table 14-30 describes the fields of the MRBL register.

Table 14-30. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	To ensure that MRBL is a multiple of 64, these bits are reserved and must be cleared.
16–25	MRBL	Maximum receive buffer length. MRBL is the number of bytes that the TSEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. TSEC can write fewer bytes to the buffer than the value set in MRBL if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBL value; therefore, user-supplied buffers must be at least as large as the MRBL.
26–31	—	To ensure that MRBL is a multiple of 64, these bits are reserved and must be cleared.

14.5.3.4.7 Receive Buffer Descriptor Pointer Register (RBPTR)

RBPTR contains the receive buffer descriptor address. Figure 14-33 shows the RBPTR register. This register takes on the value of RBASE when the RBASE register is written by software. Although not necessary in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by TSTAT[THLT].

	0	28	29		31
R	RBPTR		0	0	0
W					
Reset	0000_0000_0000_0000_0000_0000_0000_0000				
Offset	TSEC1:0x2_4384; TSEC2:0x2_5384				

Figure 14-33. RBPTR Register Definition

Table 14-31 describes the fields of the RBPTR register.

Table 14-31. RBPTR Field Descriptions

Bits	Name	Description
0–28	RBPTR	The RBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.4.8 Receive Descriptor Base Address Register (RBASE)

The RBASE register is written by the user with the RxBD base address and must be divisible by eight for the 8-byte buffer descriptors. Figure 14-34 shows the RBASE register.

	0 ... 28	29	30	31
R	RBASE	0	0	0
W				
Reset	0000_0000_0000_0000_0000_0000_0000_0000			
Offset	TSEC1:0x2_4404; TSEC2:0x2_5404			

Figure 14-34. RBASE Register Definition

Table 14-32 describes the fields of the RBASE register.

Table 14-32. RBASE Field Descriptions

Bits	Name	Description
0–28	RBASE	Receive base. RBASE defines the starting location in the memory map for the TSEC RxBD.
29–31	—	Reserved. These bits must be set to zero, to cause the value of RBASE to be a multiple of eight.

14.5.3.5 MAC Functionality

This section describes the MAC registers and provides a brief overview of the functionality that can be exercised through the use of these registers, particularly those that provide functionality not explicitly required by the IEEE 802.3 standard. All of the MAC registers are 32 bits wide.

14.5.3.5.1 Configuring the MAC

MAC configuration registers 1 and 2 provide a way to configure the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal seven bytes to some other (non-zero) value.
- Varying pad/CRC combinations—Three pad/CRC combinations are provided to handle a variety of system requirements. The most simple are frames that already have a valid FCS field. In this case, the CRC is checked and reported through the transmit statistics vector (TSV[51:0]). The other two options include appending a valid CRC, or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

14.5.3.5.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the TSEC. Half-duplex is only supported for 10 Mbps and 100 Mbps operation. Following the completion of the packet transmission the part begins timing the inter packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is now free to begin another frame transfer.

In full-duplex mode both the carrier sense (CRS) and collision (COL) indications from the PHY are ignored, but in half-duplex mode the TSEC defers to CRS and, following a carrier event, times the IPG using the non-back-to-back IPG configuration values that include support for the optional two-thirds/one-third CRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if CRS is sensed. During the final one-third of the IPG, CRS is ignored and the transmission begins once IPG is timed. The two-thirds/one-third ratio is the recommended value.

14.5.3.5.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the TSEC is sensitive to COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is comprised of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating that a collision occurred and that the start of the frame is needed for retransmission. The TSEC then backs off of the medium for a time determined by the truncated binary exponential back-off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured through the half-duplex register. However, this is non-standard behavior and its use must be carefully applied. Should any one packet experience excessive collisions, the packet is aborted. The system must flush the frame and move to the next one in line. If the system requests to send a packet while the TSEC is deferring to a carrier, the TSEC simply waits until the end of the carrier event and the timing of IPG before it honors the request.

If packet transmission attempts experience collisions, the TSEC outputs the jam sequence and waits some amount of time before retrying the packet. This amount of time is determined by a controlled randomization process called truncated binary exponential back-off. The amount of time is an integer number of slot times. The number of slot times to delay before the *n*th retransmission attempt is chosen as a uniformly-distributed random integer *r* in the range:

$$0 \leq r \leq 2^k, \text{ where } k = \min(n,10).$$

So, after the first collision, TSEC backs-off either 0 or 1 slot times. After the fifth collision, TSEC backs-off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back-off is 1024. This can be adjusted through the half-duplex register. An alternate truncation point, such as 7 for instance, can be programmed. On average, the MAC is more aggressive after seven collisions than other stations on the network.

14.5.3.5.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within TSEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time begins at the beginning of the packet, the end occurs around the 56th byte of the frame data. Slot time in 1-Gbps mode is not supported.

Full-duplex flow control is provided for in IEEE 802.3x. Currently the standard does not address flow control in half-duplex environments. Common in the industry, however, is the concept of back pressure. The TSEC implements the optional back pressure mechanism using the raise carrier method. If the system receive logic wishes to stop the reception of packets in a network-friendly way, transmit half-duplex flow

control (THDF) is set (TCTRL[THDF]). If the medium is idle, the TSEC raises carrier by transmitting preamble. Other stations on the half-duplex network then defer to the carrier.

In the event the preamble transmission happens to cause a collision, TSEC ensures the minimum 96-bit presence on the wire, then drops preamble and waits a back-off time depending on the value of the configuration bit, back pressure no back-off (half-duplex) [BPNB]. These transmitting-preamble-for-back pressure collisions are not counted. If BPNB is set, the TSEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If cleared, the TSEC adheres to the truncated BEB algorithm that allows the possibility of packets being received. This also can be detrimental in that packets can now experience excessive collisions, causing them to be dropped in the stations from which they originate. To reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, BPNB must be set.

The TSEC drops carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the TSEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. BPNB applies for any collision that occurs during the sending of this packet. TSEC does not defer while attempting to send packets while in back pressure. Again, back pressure is non-standard, yet it can be effective in reducing the flow of receive packets.

14.5.3.5.5 Controlling PHY Links

Control and status to and from the PHY is provided through the two-wire MII management interface described in IEEE 802.3u. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) are used to exercise this interface between a host processor and one or more PHY devices (including the TBI). External PHYs may only be configured using the MII management interface of TSEC1 since the interface signals (EC_MDC and EC_MDIO) are only driven by TSEC1.

The TSEC MII's registers provide the ability to perform continuous read cycles, called a scan cycle. If requested (by setting MIIMCOM[Scan Cycle]), the part performs repetitive read cycles of the PHY status register, for example. In this way, link characteristics may be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) are used to indicate availability of each read of the scan cycle to the host from MIIMSTAT[PHY Status].

Yet another parameter that can be modified through the MII registers is the length of the MII management interface preamble. After establishing that a PHY supports preamble suppression, the host may so configure the TSEC. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

14.5.3.6 MAC Registers

The MAC registers are described in the following sections.

14.5.3.6.1 MAC Configuration Register 1 (MACCFG1)

The MACCFG1 register is written by the user. Figure 14-35 shows the MACCFG1 register.

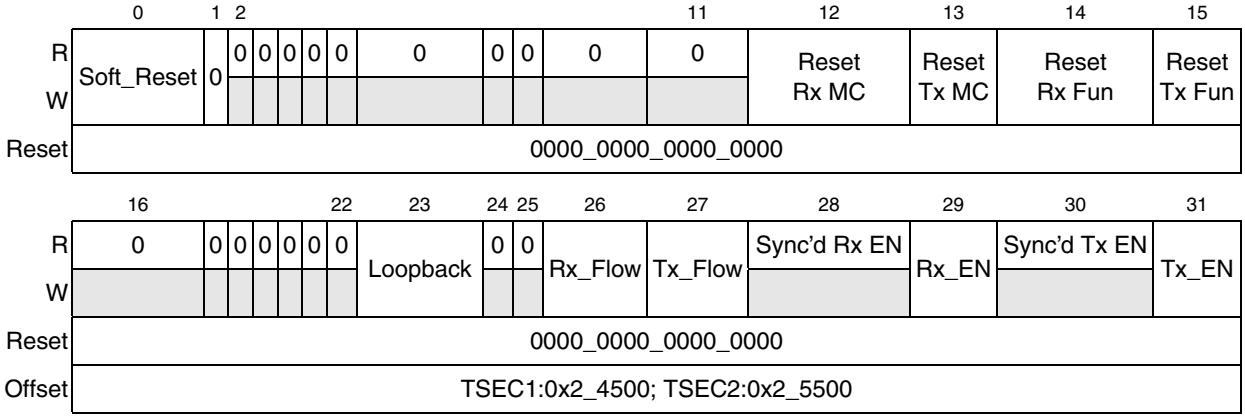


Figure 14-35. MACCFG1 Register Definition

Table 14-33 describes the fields of the MACCFG1 register.

Table 14-33. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 14.6.2.2, "Soft Reset and Reconfiguring Procedure," for more information on setting this bit. 0 Normal operation. 1 Place all modules within the MAC in reset.
1–11	—	Reserved
12	Reset Rx MC	Reset receive MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the receive MAC control block in reset. This block detects control frames and contains the pause timers.
13	Reset Tx MC	Reset transmit MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the PETMC transmit MAC control block in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.
14	Reset Rx Fun	Reset receive function block. This bit is cleared by default. 0 Normal operation. 1 Place the receive function block in reset. This block performs the receive frame protocol.
15	Reset Tx Fun	Reset transmit function block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit function block in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loopback	Loopback. This bit is cleared by default. 0 Normal operation. 1 Loopback the MAC transmit outputs to the MAC receive inputs.
24–25	—	Reserved

Table 14-33. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
26	Rx_Flow	Receive flow. This bit is cleared by default. 0 The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control detects and acts on PAUSE flow control frames.
27	Tx_Flow	Transmit flow. This bit is cleared by default. 0 The transmit MAC control may not send PAUSE flow control frames if requested by the system. 1 The transmit MAC control may send PAUSE flow control frames if requested by the system.
28	Sync'd Rx EN	Receive enable synchronized to the receive stream (Read-only) 0 Frame reception is not enabled. 1 Frame reception is enabled.
29	Rx_EN	Receive enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set). 0 The MAC may not receive frames from the PHY. 1 The MAC may receive frames from the PHY.
30	Sync'd Tx EN	Transmit enable synchronized to the transmit stream (Read-only) 0 Frame transmission is not enabled. 1 Frame transmission is enabled.
31	Tx_EN	Transmit enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful transmit stop interrupt (IEVENT[GTSC] is set). 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

14.5.3.6.2 MAC Configuration Register 2 (MACCFG2)

The MACCFG2 register is written by the user. Figure 14-36 shows the MACCFG2 register.

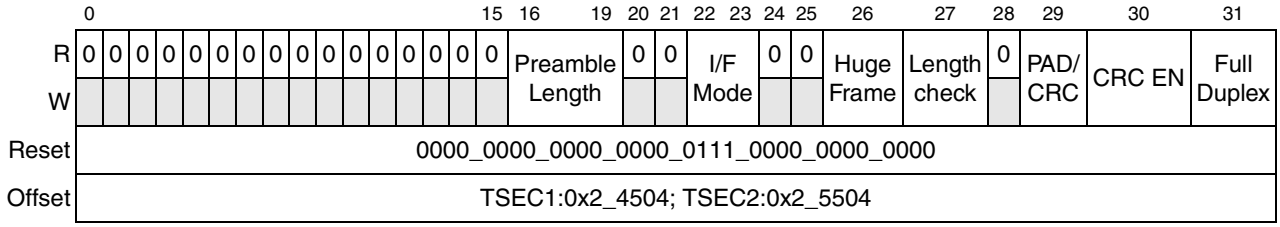


Figure 14-36. MACCFG2 Register Definition

Table 14-34 describes the fields of the MACCFG2 register.

Table 14-34. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble Length	Determines the length in bytes of the preamble field of the packet. Its default is 0x7. A preamble length of 0 is not supported.
20–21	—	Reserved

Table 14-34. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
22–23	I/F Mode	Determines the type of interface to which the MAC is connected. Its default is 00. 00 Reserved 01 Nibble mode (MII) 10 Byte mode (GMII/TBI) 11 Reserved
24–25	—	Reserved
26	Huge Frame	Huge frame enable. Cleared by default. 0 Limit the length of frames to the MAXIMUM FRAME LENGTH value. 1 Frames longer than the MAXIMUM FRAME LENGTH may be transmitted and received.
27	Length check	Length check. This bit is cleared by default. 0 No length field checking is performed. 1 The MAC checks the frame's length field to ensure it matches the actual data field length.
28	—	Reserved
29	PAD/CRC	Pad and append CRC. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame regardless of padding requirement.
30	CRC EN	CRC enable. If the configuration bit PAD/CRC or the per-packet PAD/CRC bit is set, CRC EN is ignored. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a valid CRC. 1 The MAC appends a CRC on all frames. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC.
31	FULL DUPLEX	Full duplex configure. This bit is cleared by default. 0 The MAC to operate in half-duplex mode only. 1 The MAC operates in full-duplex mode.

14.5.3.6.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user. Figure 14-37 shows the IPGIFG register.

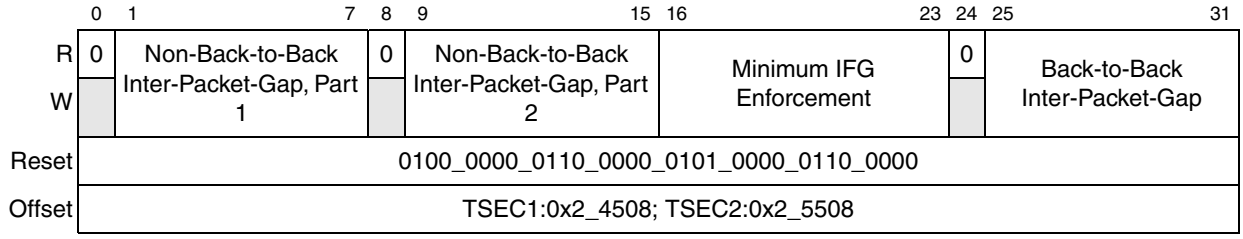


Figure 14-37. IPGIFG Register Definition

Table 14-35 describes the fields of the IPGIFG register.

Table 14-35. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-Packet-Gap, Part 1	Programmable field representing the optional carrier Sense window referenced in IEEE 802.3/4.2.3.2.1 ‘carrier deference’. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision; thus, ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-third guideline.
8	—	Reserved
9–15	Non-Back-to-Back Inter-Packet-Gap, Part 2	Programmable field representing the non-back-to-back inter-packet-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	Programmable field representing the minimum number of bits of IFG to enforce between frames. A frame is dropped whose IFG is less than that programmed. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	Programmable field representing the IPG between back-to-back packets. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

14.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 14-38 shows the HAFDUP register.

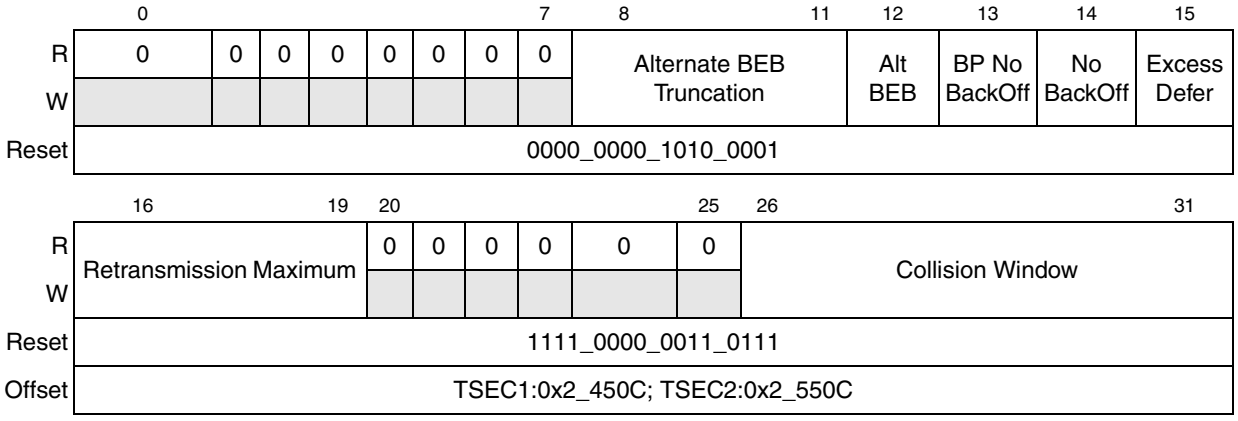


Figure 14-38. Half-Duplex Register Definition

Table 14-36 describes the fields of the HAFDUP register.

Table 14-36. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while alternate binary exponential back-off enable is set. The value programmed is substituted for the Ethernet standard value of ten. Its default is 0xA.
12	Alt BEB	Alternate binary exponential back-off. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential back-off rule. 1 The Tx MAC uses the alternate binary exponential back-off truncation setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 210 as the maximum back-off time.
13	BP No BackOff	Back pressure no back-off. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back-off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.
14	No BackOff	No back-off. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back-off rule. 1 The Tx MAC immediately re-transmits following a collision.
15	Excess Defer	Excessively deferred. This bit is set by default. 0 The Tx MAC aborts the transmission of a packet that is excessively deferred. 1 The Tx MAC allows the transmission of a packet that is excessively deferred.
16–19	Retransmission Maximum	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d). Its default value is 0xF.
20–25	—	Reserved
26–31	Collision Window	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

14.5.3.6.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 14-39 shows the MAXFRM register.

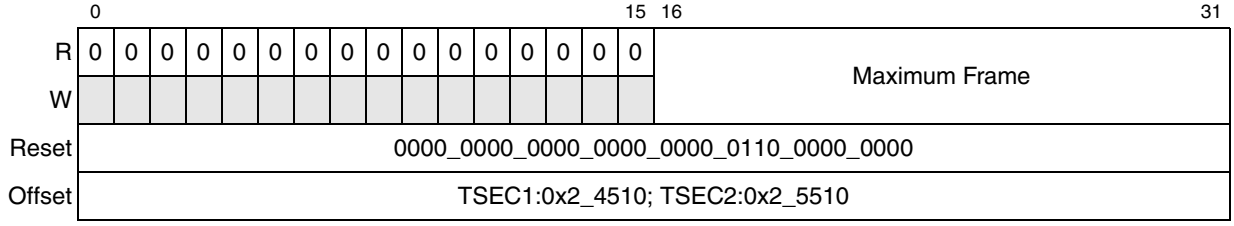


Figure 14-39. Maximum Frame Length Register Definition

Table 14-37 describes the fields of the MAXFRM register.

Table 14-37. MAXFRM Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	By default this field is set to 0x0600 (1536d). It sets the maximum frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].)

14.5.3.6.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 14-40 shows the MIIMCFG register.

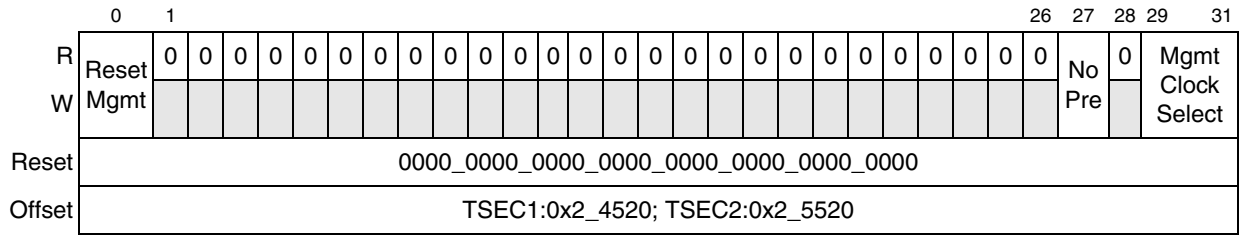


Figure 14-40. MII Management Configuration Register Definition

Table 14-38 describes the fields of the MIIMCFG register.

Table 14-38. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII Mgmt to perform management read/write cycles if requested. 1 Reset the MII Mgmt.
1–26	—	Reserved
27	No Pre	Preamble suppress. This bit is cleared by default. 0 The MII Mgmt performs management read/write cycles with 32 clocks of preamble. 1 The MII Mgmt suppresses preamble generation and reduces the management cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.
28	—	Reserved
29–31	Mgmt Clock Select	This field determines the clock frequency of the Mgmt Clock (EC_MDC). Its default value is 000. The source clock frequency is equal to the core complex bus clock divided first by eight and then further divided by the following value: 000 source clock divided by 4 001 source clock divided by 4 010 source clock divided by 6 011 source clock divided by 8 100 source clock divided by 10 101 source clock divided by 14 110 source clock divided by 20 111 source clock divided by 28

14.5.3.6.7 MII Management Command Register (MIIMCOM)

The MIIMCOM register is written by the user. Figure 14-41 shows the MIIMCOM register.

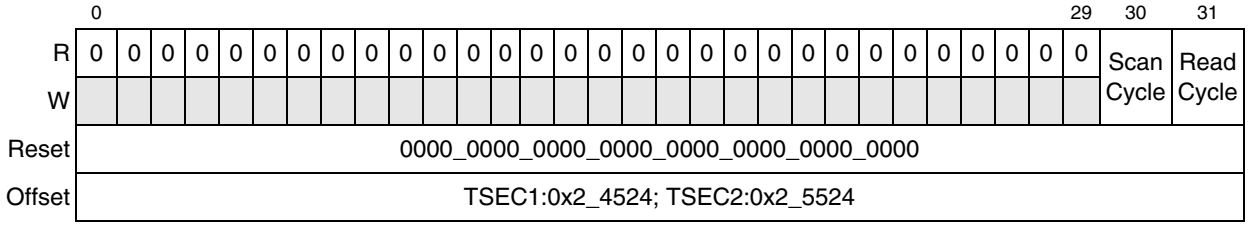


Figure 14-41. MIIMCOM Register Definition

Table 14-39 describes the fields of the MIIMCOM register.

Table 14-39. MIIMCOM Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	Scan Cycle	Scan cycle. This bit is cleared by default. 0 Normal operation. 1 The MII management continuously performs read cycles. This is useful for polling a PHY register, for example, monitoring link fails. Data is returned in register MIIMSTAT[PHY Status]. IEVENT[MMRD] is set once a read has been completed.
31	Read Cycle	Read cycle. This bit is cleared by default but is not self-clearing once set. 0 Normal operation. 1 The MII management performs a single read cycle upon the transition of this bit from 0 to 1 using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The 0 to 1 transition of this bit also causes the MIIMIND[BUSY] bit to be set. The read is complete when the MIIMIND[BUSY] bit clears. Data is returned in register MIIMSTAT[PHY Status]. IEVENT[MMRD] is also set once the read has been completed.

14.5.3.6.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 14-42 shows the MIIMADD register.

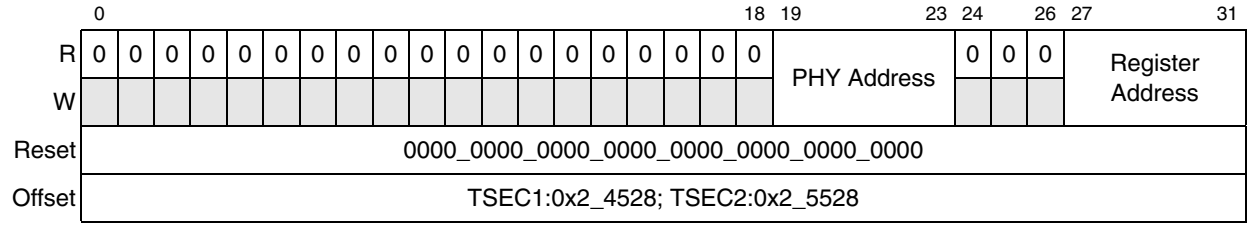


Figure 14-42. MIIMADD Register Definition

Table 14-40 describes the fields of the MIIMADD register.

Table 14-40. MIIMADD Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of management cycles. Up to 31 PHYs can be addressed. At least one PHY address is reserved for the TBI PHY address as programmed in the TBIPA register (Section 14.5.3.1.8, “TBI Physical Address Register (TBIPA),” Figure 14-13). The default of the TBI PHY address is 0x00. The user must be sure to assign a physical address to the TBI so as to not conflict with the external PHY physical address as discussed in the register initialization steps in Section 14.7, “Initialization/Application Information.”
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of management cycles. Up to 32 registers can be accessed. Its default value is 0x00.

14.5.3.6.9 MII Management Control Register (MIIMCON)

The MIIMCON register is written by the user. Figure 14-43 shows the MIIMCON register.

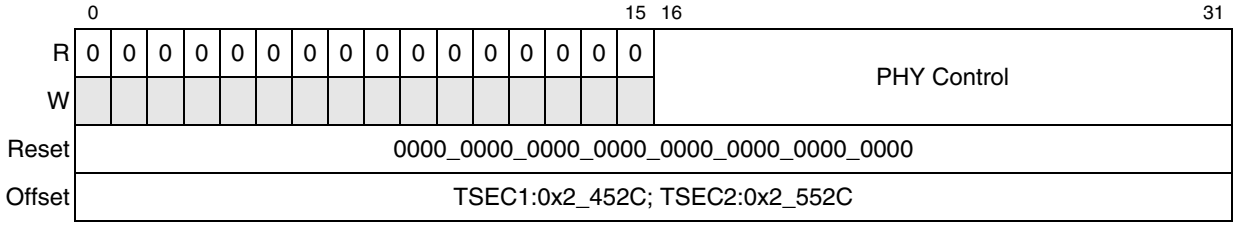


Figure 14-43. MII Management Control Register Definition

Table 14-41 describes the fields of the MIIMCON register.

Table 14-41. MIIMCON Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Control	If written, an MII Mgmt write cycle is performed using this 16-bit data, the pre-configured PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). Its default value is 0x0000. IEVENT[MMWR] is set once the write has been completed.

14.5.3.6.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT, shown in Figure 14-44, is read-only by the user.

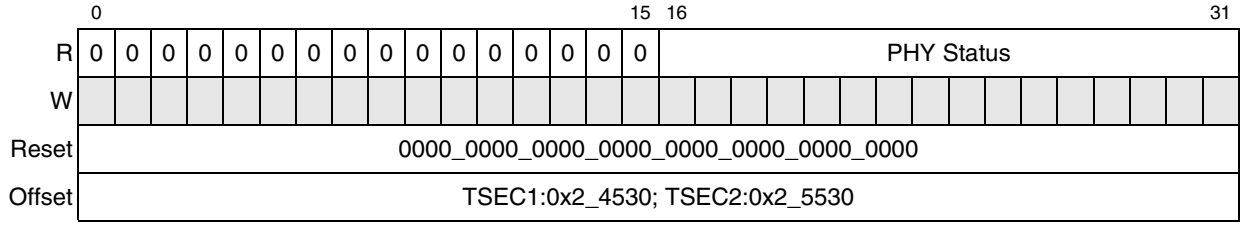


Figure 14-44. MIIMSTAT Register Definition

Table 14-42 describes the fields of the MIIMSTAT register.

Table 14-42. MIIMSTAT Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Status	Following an MII Mgmt read cycle, the 16-bit data can be read from this location. Its default value is 0x0000.

14.5.3.6.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 14-45 shows the MIIMIND register.

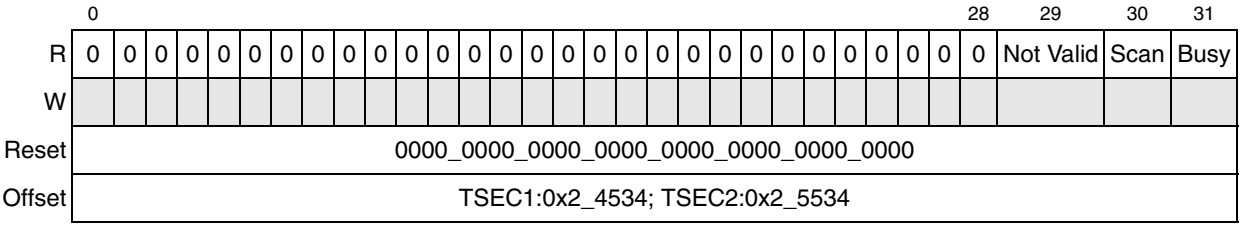


Figure 14-45. MII Management Indicator Register Definition

Table 14-43 describes the fields of the MIIMIND register.

Table 14-43. MIIMIND Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	Not Valid	Not valid 0 MII Mgmt read cycle has completed and the read data is valid. 1 MII Mgmt read cycle has not completed and the read data is not yet valid.

Table 14-43. MIIMIND Field Descriptions (continued)

Bits	Name	Description
30	Scan	Scan in progress 0 A scan operation (continuous MII Mgmt read cycles) is not in progress. 1 A scan operation (continuous MII Mgmt read cycles) is in progress.
31	Busy	Busy 0 MII Mgmt block is not currently performing an MII Mgmt read or write cycle. 1 MII Mgmt block is currently performing an MII Mgmt read or write cycle. IEVENT[MMRD] or IEVENT[MMWR] is set once the read or write, respectively, has been completed.

14.5.3.6.12 Interface Status Register (IFSTAT)

The IFSTAT register is written by the user. Figure 14-46 shows the IFSTAT register.

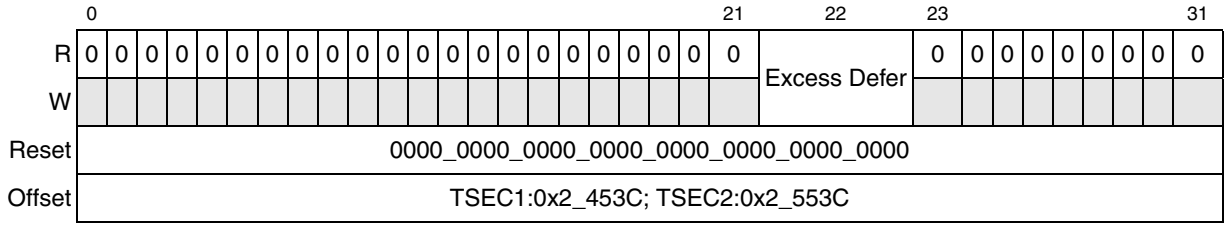


Figure 14-46. Interface Status Register Definition

Table 14-44 describes the fields of the IFSTAT register.

Table 14-44. IFSTAT Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	Excess Defer	Excessive transmission defer. This bit latches high and is cleared when read. This bit is cleared by default. 0 Normal operation. 1 The MAC excessively defers a transmission.
23–31	—	Reserved

14.5.3.6.13 Station Address Register Part 1 (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 14-47 shows the MACSTNADDR1 register. The value of the station address written by the user into MACSTNADDR1 and MACSTNADDR2 is byte-reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678ABCD, perform a write to MACSTNADDR1 of 0xCDAB7856, and to MACSTNADDR2 of 0x34120000. When the user reads MACSTNADDR1, 0xCDAB7856 is returned. A read of MACSTNADDR2 returns a value of 0x34120000. Note, the I/G and U/L bits of the frame's DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

	0–7	8–15	16–23	24–31
R W	Station Address, 6th Octet	Station Address, 5th Octet	Station Address, 4th Octet	Station Address, 3rd Octet
Reset	0000_0000_0000_0000_0000_0000_0000_0000			
Offset	TSEC1:0x2_4540; TSEC2:0x2_5540			

Figure 14-47. Station Address Part 1 Register Definition

Table 14-45 describes the fields of the MACSTNADDR1 register.

Table 14-45. MACSTNADDR1 Field Descriptions

Bits	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 0x00.
8–15	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet (station address bits 32–39) defaults to a value of 0x00.
16–23	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet (station address bits 24–31) defaults to a value of 0x00.
24–31	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet (station address bits 16–23) defaults to a value of 0x00.

14.5.3.6.14 Station Address Register Part 2 (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 14-48 shows the MACSTNADDR2 register. Note, the I/G and U/L bits of the frame's DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

	0–7	8–15	16														31
R	Station Address, 2nd Octet	Station Address, 1st Octet	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																	
Reset	0000_0000_0000_0000_0000_0000_0000_0000																
Offset	TSEC1:0x2_4544; TSEC2:0x2_5544																

Figure 14-48. Station Address Part 2 Register Definition

Table 14-46 describes the fields of the MACSTNADDR2 register.

Table 14-46. MACSTNADDR2 Field Descriptions

Bits	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 0x00.
8–15	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet (station address bits 0–7) defaults to a value of 0x00.
16–31	—	Reserved

14.5.3.7 MIB Registers

This section describes the MIB registers.

The TSEC MSTAT module has 37 separate statistics counters, which simply count or accumulate statistical events that occur as packets are transmitted and received. These counters support RMON MIB group 1, RMON MIB group 2, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB.

The detection of one or more of these statistical events triggers the MSTAT module to update its statistics counters. These counters are stored in internal data registers. The user may access the internal data registers at any time. An interrupt can be generated upon any one counter's rollover condition through a carry interrupt output from the MSTAT. Each counter's rollover condition can be discreetly masked from causing an interrupt by internal masking registers. In addition, each individual counter value may be reset on read access, or all counters may be simultaneously reset by assertion of an external module input signal. Figure 14-49 shows the TR64 register.

14.5.3.7.1 Transmit and Receive 64-Byte Frame Counter Register (TR64)

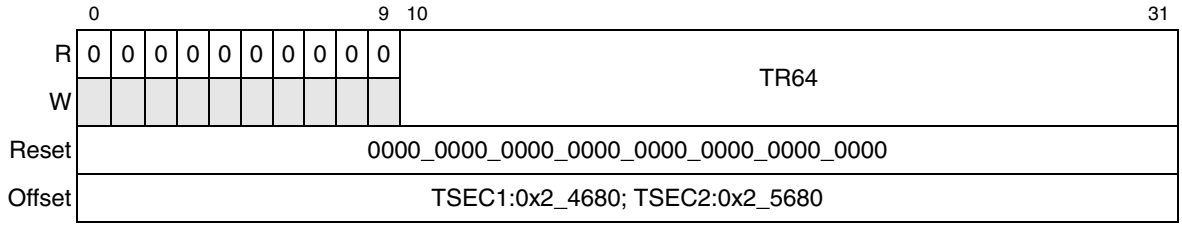


Figure 14-49. Transmit and Receive 64-Byte Frame Register Definition

Table 14-47 describes the fields of the TR64 register.

Table 14-47. TR64 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR64	Transmit and receive 64-byte frame counter—Increment for each good or bad frame transmitted and received which is 64 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.2 Transmit and Receive 65- to 127-Byte Frame Counter Register (TR127)

Figure 14-50 shows the TR127 register.

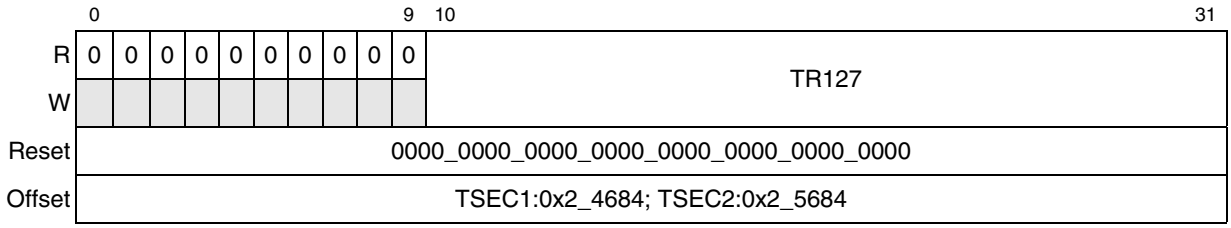


Figure 14-50. Transmit and Receive 65- to 127-Byte Frame Register Definition

Table 14-48 describes the fields of the TR127 register.

Table 14-48. TR127 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR127	Transmit and receive 65- to 127-byte frame counter—Increment for each good or bad frame transmitted and received which is 65 to 127 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.3 Transmit and Receive 128- to 255-Byte Frame Counter Register (TR255)

Figure 14-51 shows the TR255 register.

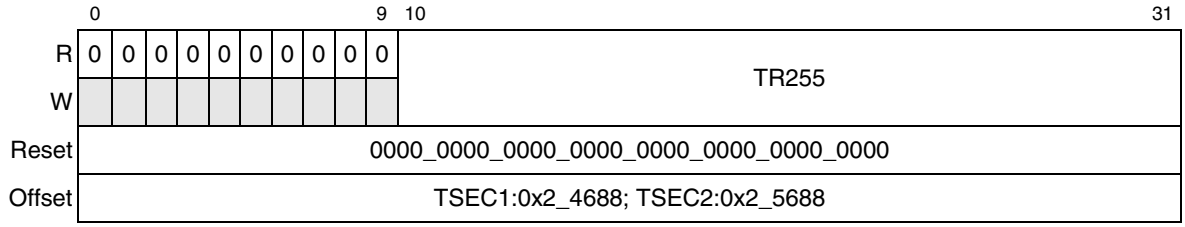


Figure 14-51. Transmit and Receive 128- to 255-Byte Frame Register Definition

Table 14-49 describes the fields of the TR255 register.

Table 14-49. TR255 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR255	Transmit and receive 128- to 255-byte frame counter—Increments for each good or bad frame transmitted and received which is 128 to 255 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.4 Transmit and Receive 256- to 511-Byte Frame Counter Register (TR511)

Figure 14-52 shows the TR511 register.

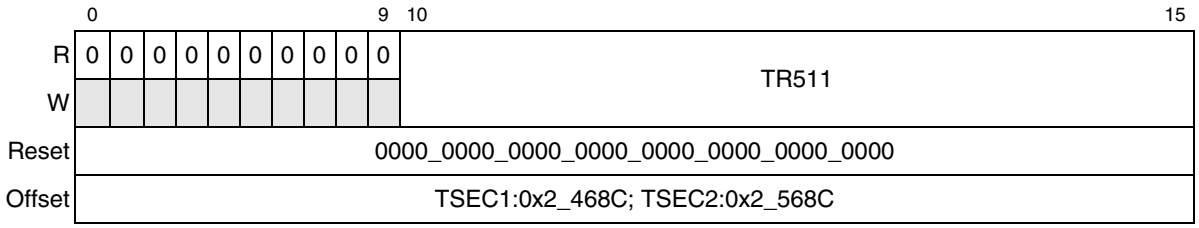


Figure 14-52. Transmit and Receive 256- to 511-Byte Frame Register Definition

Table 14-50 describes the fields of the TR511 register.

Table 14-50. TR511 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR511	Increments for each good or bad frame transmitted and received which is 256 to 511 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.5 Transmit and Receive 512- to 1023-Byte Frame Counter Register (TR1K)

Figure 14-53 shows the TR1K register.

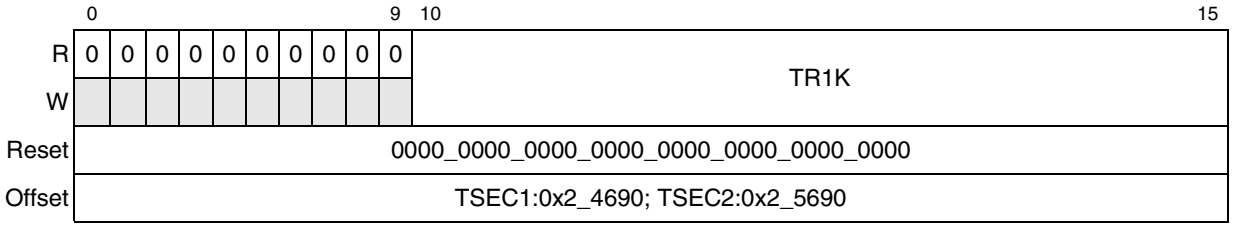


Figure 14-53. Transmit and Receive 512- to 1023-Byte Frame Register Definition

Table 14-51 describes the fields of the TR1K register.

Table 14-51. TR1K Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR1K	Increments for each good or bad frame transmitted and received which is 512 to 1023 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.6 Transmit and Receive 1024- to 1518-Byte Frame Counter Register (TRMAX)

Figure 14-54 shows the TRMAX register.

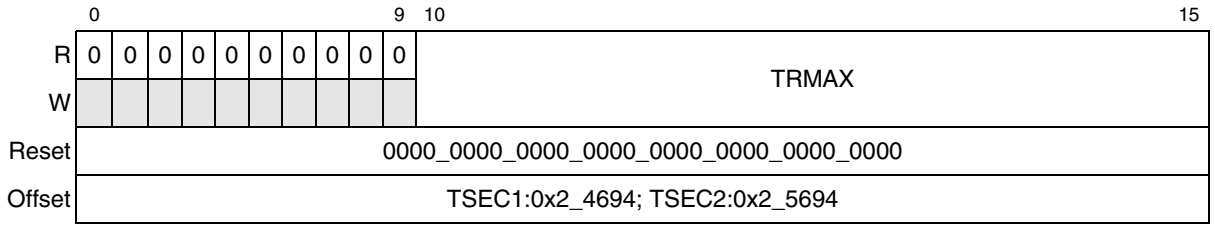


Figure 14-54. Transmit and Receive 1024- to 1518-Byte Frame Register Definition

Table 14-52 describes the fields of the TRMAX register.

Table 14-52. TRMAX Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TRMAX	Increments for each good or bad frame transmitted and received which is 1024 to 1518 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.7 Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter Register (TRMGV)

Figure 14-55 shows the TRMGV register.

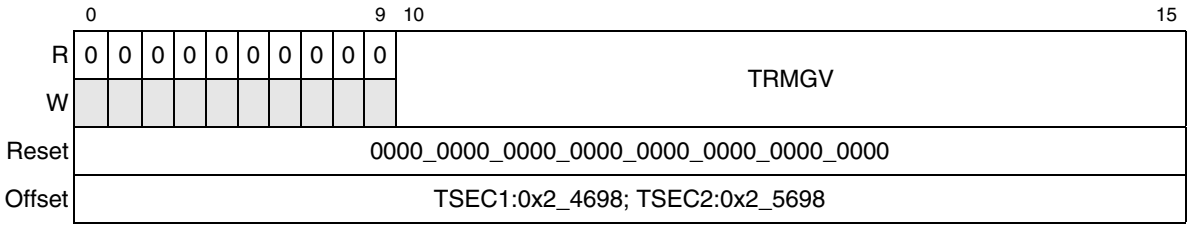


Figure 14-55. Transmit and Receive 1519- to 1522-Byte VLAN Frame Register Definition

Table 14-53 describes the fields of the TRMGV register.

Table 14-53. TRMGV Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TRMGV	Increments for each good or bad frame transmitted and received which is 1519 to 1522 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes)

14.5.3.7.8 Receive Byte Counter Register (RBYT)

Figure 14-56 shows the RBYT register.

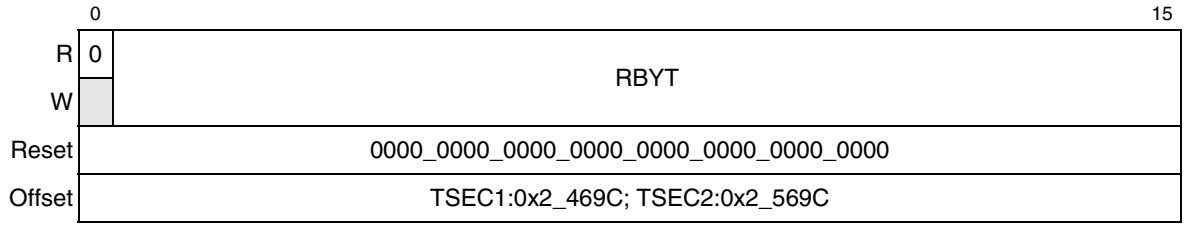


Figure 14-56. Receive Byte Counter Register Definition

Table 14-54 describes the fields of the RBYT register.

Table 14-54. RBYT Field Descriptions

Bits	Name	Description
0	—	Reserved
1–31	RBYT	Receive byte counter. Increments by the byte count of frames received, including those in bad packets, excluding preamble and SFD but including FCS bytes.

14.5.3.7.9 Receive Packet Counter Register (RPKT)

Figure 14-57 shows the RPKT register.

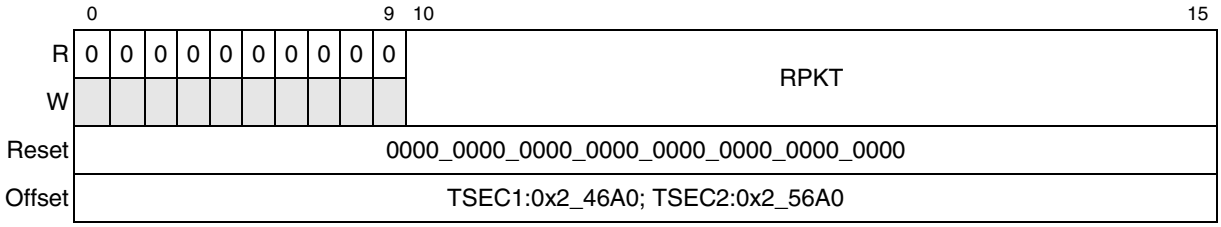


Figure 14-57. Receive Packet Counter Register Definition

Table 14-55 describes the fields of the RPKT register.

Table 14-55. RPKT Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10-31	RPKT	Receive packet counter. Increments for each frame received packet (including bad packets, all unicast, broadcast, and multicast packets).

14.5.3.7.10 Receive FCS Error Counter Register (RFCS)

Figure 14-58 shows the RFCS register.

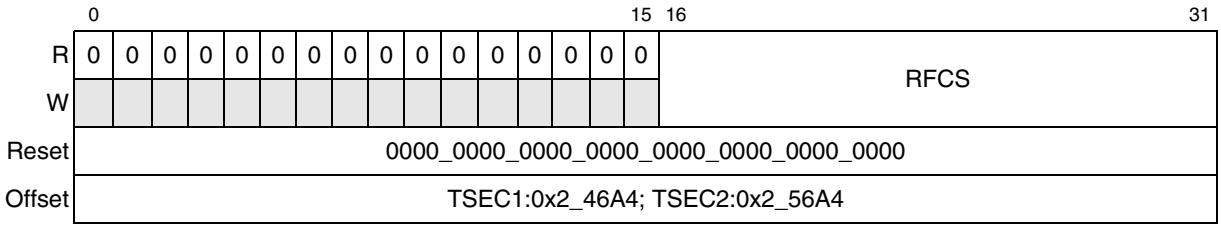


Figure 14-58. Receive FCS Error Counter Register Definition

Table 14-56 describes the fields of the RFCS register.

Table 14-56. RFCS Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFCS	Receive FCS error counter. Increments for each frame received that has an integral 64 to 1518 length and contains a frame check sequence error.

14.5.3.7.11 Receive Multicast Packet Counter Register (RMCA)

Figure 14-59 shows the RMCA register.

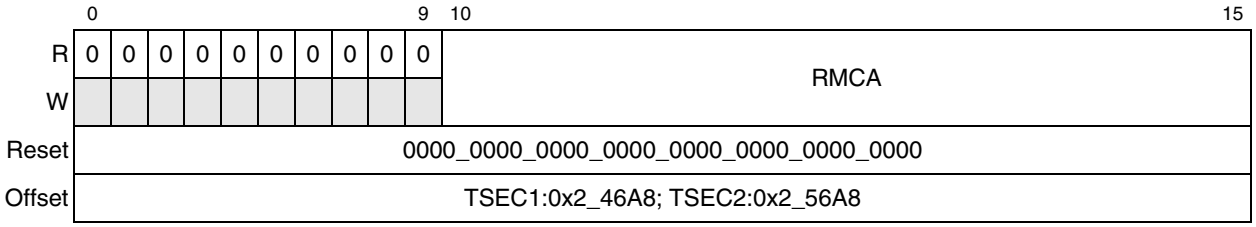


Figure 14-59. Receive Multicast Packet Counter Register Definition

Table 14-57 describes the fields of the RMCA register.

Table 14-57. RMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RMCA	Receive multicast packet counter. Increments for each multicast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding broadcast frames. This count does not include range/length errors.

14.5.3.7.12 Receive Broadcast Packet Counter Register (RBCA)

Figure 14-59 shows the RBCA register.

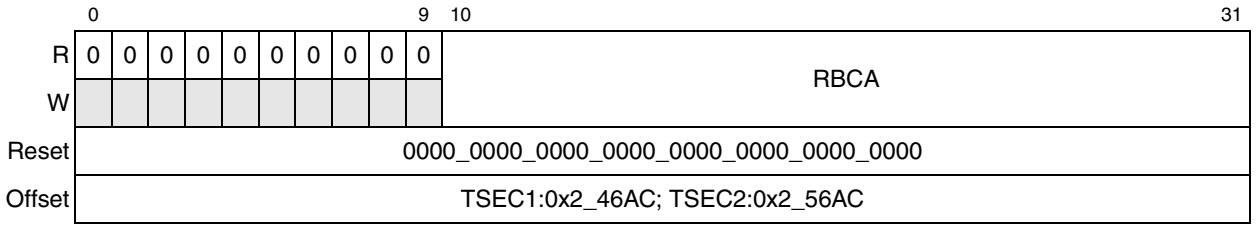


Figure 14-60. Receive Broadcast Packet Counter Register Definition

Table 14-58 describes the fields of the RBCA register.

Table 14-58. RBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RBCA	Receive broadcast packet counter. Increments for each broadcast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding multicast frames. Does not include range/length errors.

14.5.3.7.13 Receive Control Frame Packet Counter Register (RXCF)

Figure 14-61 shows the RXCF register.

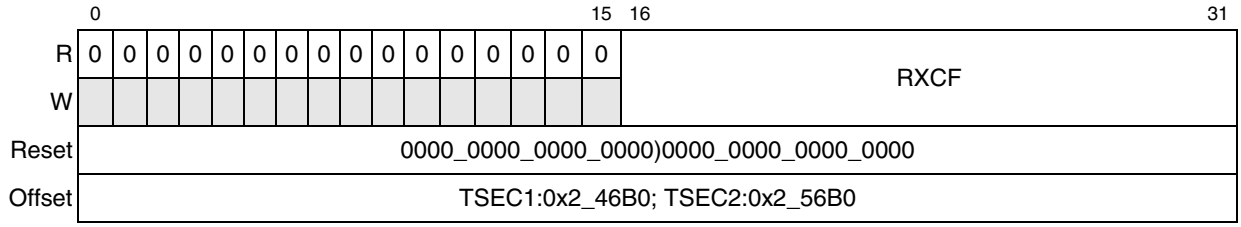


Figure 14-61. Receive Control Frame Packet Counter Register Definition

Table 14-59 describes the fields of the RXCF register.

Table 14-59. RXCF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXCF	Receive control frame packet counter. Increments for each MAC control frame received (both pause control frames and control frames unsupported by IEEE).

14.5.3.7.14 Receive Pause Frame Packet Counter Register (RXPF)

Figure 14-62 shows the RXPF register.

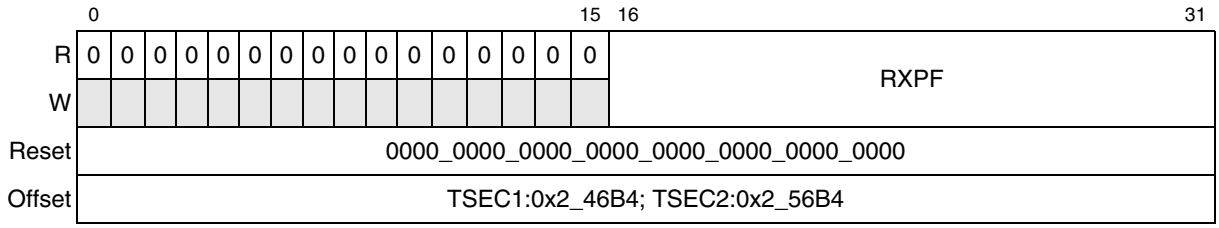


Figure 14-62. Receive Pause Frame Packet Counter Register Definition

Table 14-60 describes the fields of the RXPF register.

Table 14-60. RXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXPF	Receive PAUSE frame packet counter. Increments each time a valid PAUSE MAC control frame is received.

14.5.3.7.15 Receive Unknown Opcode Packet Counter Register (RXUO)

Figure 14-63 shows the RXUO register.

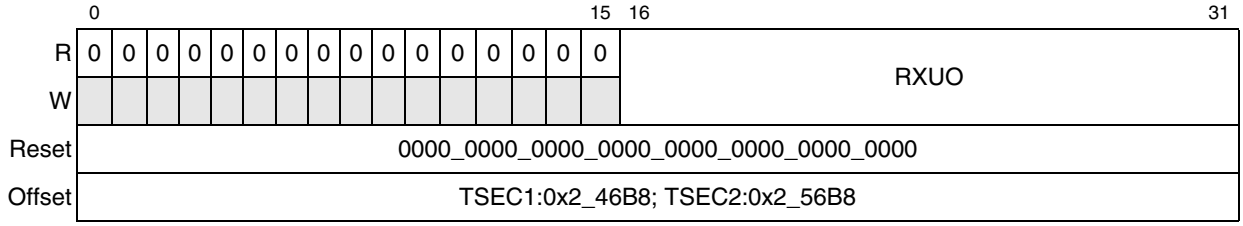


Figure 14-63. Receive Unknown Opcode Packet Counter Register Definition

Table 14-61 describes the fields of the RXUO register.

Table 14-61. RXUO Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXUO	Receive unknown opcode counter. Increments each time a MAC control frame is received which contains an opcode other than a PAUSE.

14.5.3.7.16 Receive Alignment Error Counter Register (RALN)

Figure 14-64 shows the RALN register.

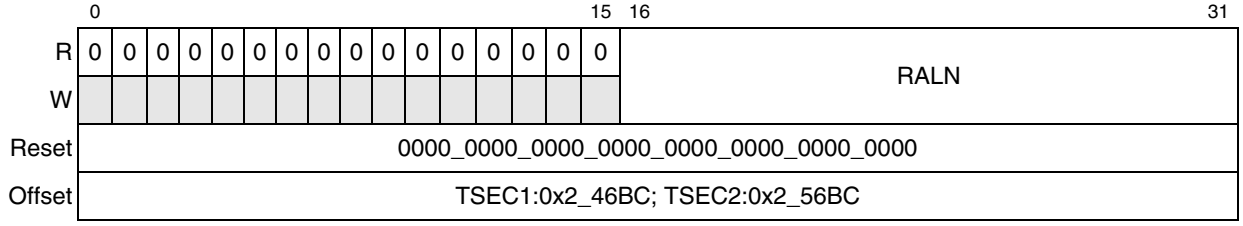


Figure 14-64. Receive Alignment Error Counter Register Definition

Table 14-62 describes the fields of the RALN register.

Table 14-62. RALN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RALN	Receive alignment error counter. Increments for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.

14.5.3.7.17 Receive Frame Length Error Counter Register (RFLR)

Figure 14-65 shows the RFLR register.

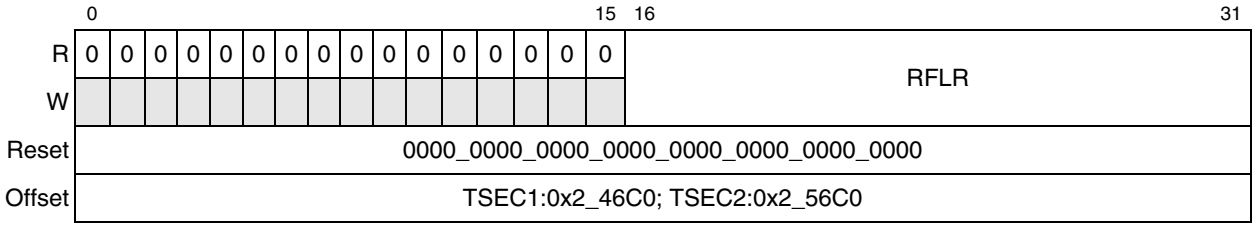


Figure 14-65. Receive Frame Length Error Counter Register Definition

Table 14-63 describes the fields of the RFLR register.

Table 14-63. RFLR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFLR	Receive frame length error counter. Increments for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46 –1500 bytes). The counter does not increment if the length field is not a valid 802.3 length, such as an ethertype value.

14.5.3.7.18 Receive Code Error Counter Register (RCDE)

Figure 14-66 shows the RCDE register.

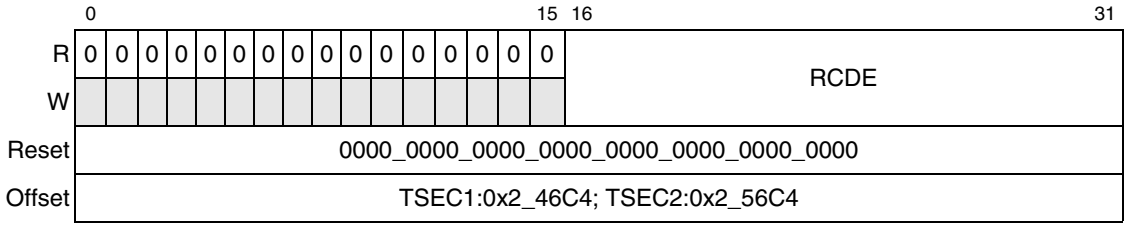


Figure 14-66. Receive Code Error Counter Register Definition

Table 14-64 describes the fields of the RCDE register.

Table 14-64. RCDE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCDE	Receive code error counter. Increments each time a valid carrier is present and at least one invalid data symbol is detected.

14.5.3.7.19 Receive Carrier Sense Error Counter Register (RCSE)

Figure 14-67 shows the RCSE register.

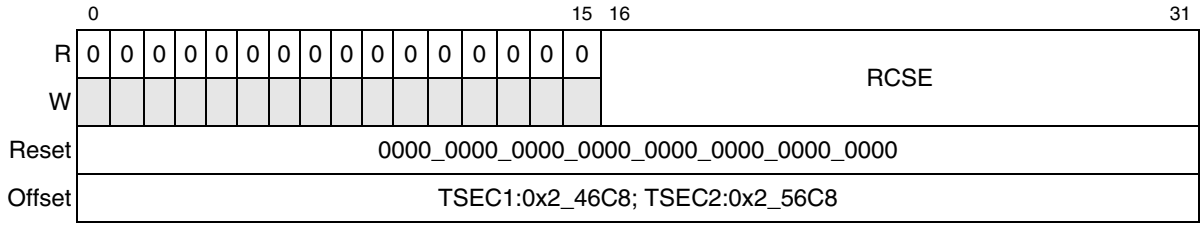


Figure 14-67. Receive Carrier Sense Error Counter Register Definition

Table 14-65 describes the fields of the RCSE register.

Table 14-65. RCSE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCSE	Receive false carrier counter. Increments each time a false carrier is detected during idle, as defined by a 1 on TSEC*n*_RX_ER and an 0xE on TSEC*n*_RXD. The event is reported along with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.

14.5.3.7.20 Receive Undersize Packet Counter Register (RUND)

Figure 14-68 shows the RUND register.

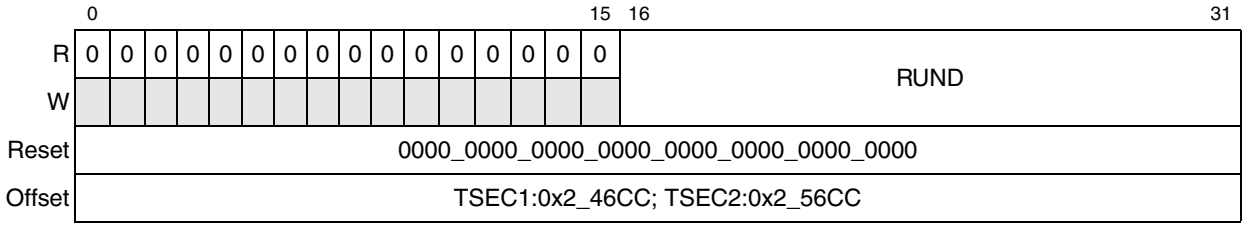


Figure 14-68. Receive Undersize Packet Counter Register Definition

Table 14-66 describes the fields of the RUND register.

Table 14-66. RUND Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RUND	Receive undersize packet counter. Increments each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This count does not include range length errors.

14.5.3.7.21 Receive Oversize Packet Counter Register (ROVR)

Figure 14-69 shows the ROVR register.

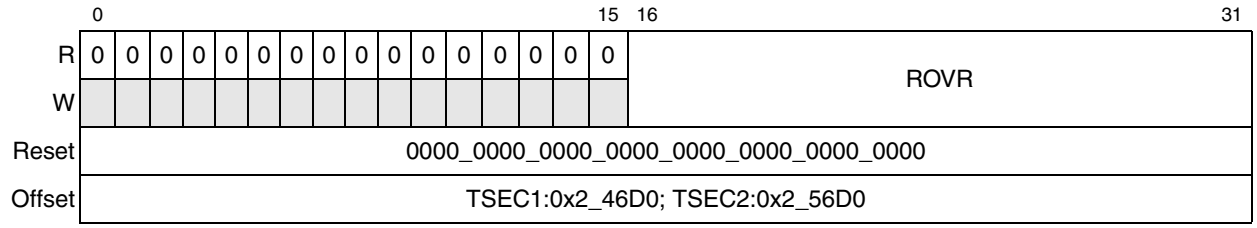


Figure 14-69. Receive Oversize Packet Counter Register Definition

Table 14-67 describes the fields of the ROVR register.

Table 14-67. ROVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	ROVR	Receive oversize packet counter. Increments each time a frame is received which exceeded 1518 (non VLAN) or 1522 (VLAN) and contains a valid FCS and was otherwise well formed. This count does not include range length errors.

14.5.3.7.22 Receive Fragments Counter Register (RFRG)

Figure 14-70 shows the RFRG register.

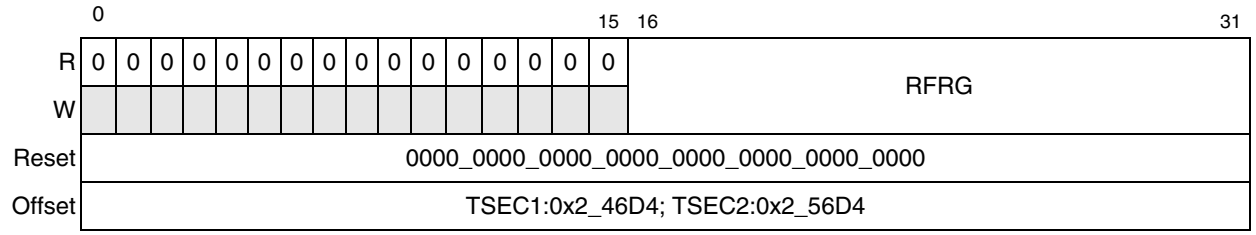


Figure 14-70. Receive Fragments Counter Register Definition

Table 14-68 describes the fields of the RFRG register.

Table 14-68. RFRG Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFRG	Receive fragments counter. Increments for each frame received which is less than 64 bytes in length and contains an invalid FCS. This includes integral and non-integral lengths.

14.5.3.7.23 Receive Jabber Counter Register (RJBR)

Figure 14-71 shows the RJBR register.

	0															15	16	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RJBR	
W																		
Reset	0000_0000_0000_0000_0000_0000_0000_0000																	
Offset	TSEC1:0x2_46D8; TSEC2:0x2_56D8																	

Figure 14-71. Receive Jabber Counter Register Definition

Table 14-69 describes the fields of the RJBR register.

Table 14-69. RJBR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16-31	RJBR	Receive jabber counter. Increments for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contain an invalid FCS. This includes alignment errors.

14.5.3.7.24 Receive Dropped Packet Counter Register (RDRP)

Figure 14-72 shows the RDRP register.

	0															15	16	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RDRP	
W																		
Reset	0000_0000_0000_0000_0000_0000_0000_0000																	
Offset	TSEC1:0x2_46DC; TSEC2:0x2_56DC																	

Figure 14-72. Receive Dropped Packet Counter Register Definition

Table 14-70 describes the fields of the RDRP register.

Table 14-70. RDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RDRP	Receive dropped packets counter. Increments for frames received which are streamed to system but are later dropped due to lack of system resources.

14.5.3.7.25 Transmit Byte Counter Register (TBYT)

Figure 14-73 shows the TBYT register.

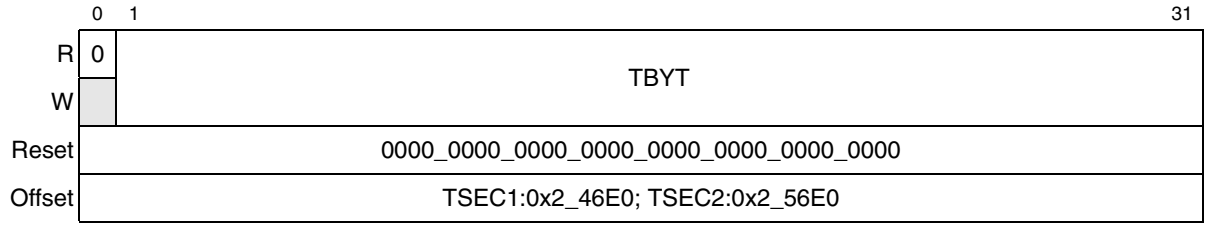


Figure 14-73. Transmit Byte Counter Register Definition

Table 14-71 describes the fields of the TBYT register.

Table 14-71. TBYT Field Descriptions

Bits	Name	Description
0	—	Reserved
1–31	TBYT	Transmit byte counter. Increments by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes. This counter does not count if the frame is truncated.

14.5.3.7.26 Transmit Packet Counter Register (TPKT)

Figure 14-74 shows the TPKT register.

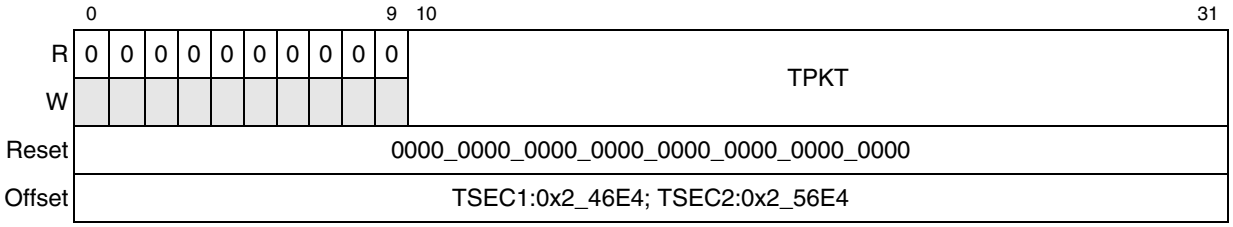


Figure 14-74. Transmit Packet Counter Register Definition

Table 14-72 describes the fields of the TPKT register.

Table 14-72. TPKT Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TPKT	Transmit packet counter. Increments for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all unicast, broadcast, and multicast packets).

14.5.3.7.27 Transmit Multicast Packet Counter Register (TMCA)

Figure 14-75 shows the TMCA register.

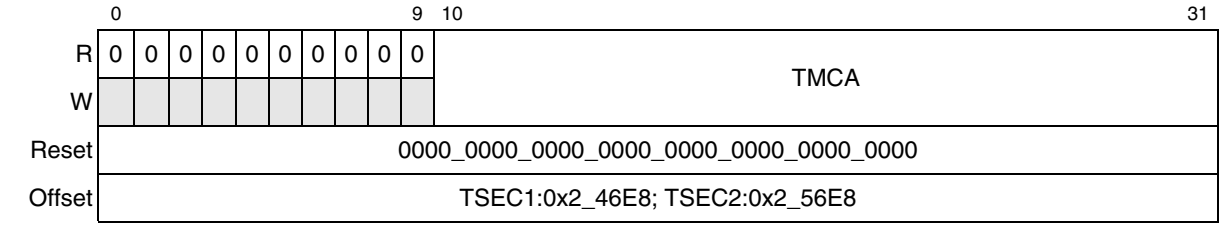


Figure 14-75. Transmit Multicast Packet Counter Register Definition

Table 14-73 describes the fields of the TMCA register.

Table 14-73. TMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TMCA	Transmit multicast packet counter. Increments for each multicast valid frame transmitted (excluding broadcast frames).

14.5.3.7.28 Transmit Broadcast Packet Counter Register (TBCA)

Figure 14-76 shows the TBCA register.

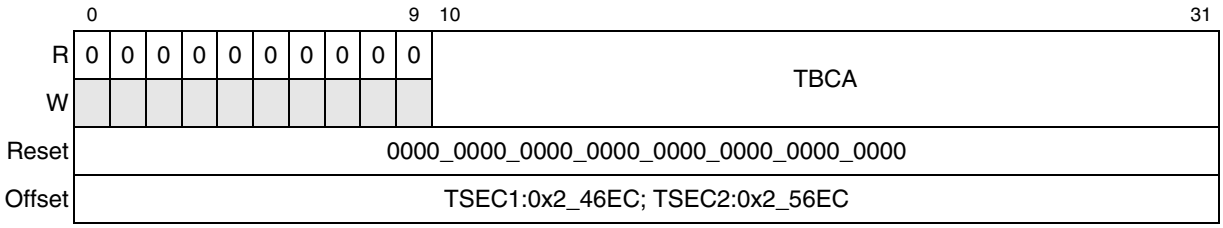


Figure 14-76. Transmit Broadcast Packet Counter Register Definition

Table 14-74 describes the fields of the TBCA register.

Table 14-74. TBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TBCA	Transmit broadcast packet counter. Increments for each broadcast frame transmitted (excluding multicast frames).

14.5.3.7.29 Transmit Pause Control Frame Counter Register (TXPF)

Figure 14-77 shows the TXPF register.

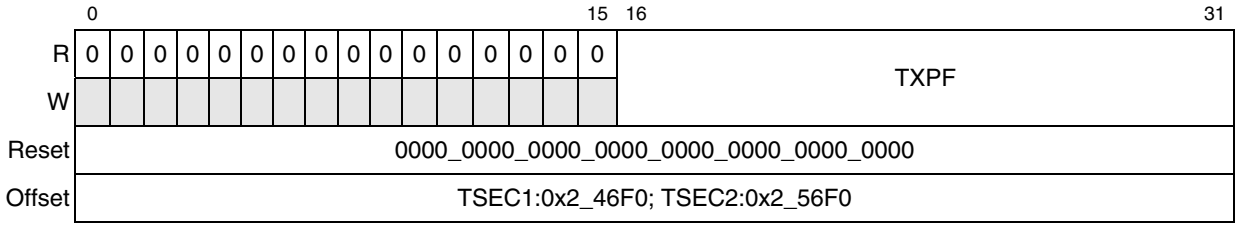


Figure 14-77. Transmit Pause Control Frame Counter Register Definition

Table 14-75 describes the fields of the TXPF register.

Table 14-75. TXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TXPF	Transmit PAUSE frame packet counter. Increments each time a valid PAUSE MAC control frame is transmitted.

14.5.3.7.30 Transmit Deferral Packet Counter Register (TDFR)

Figure 14-78 shows the TDFR register.

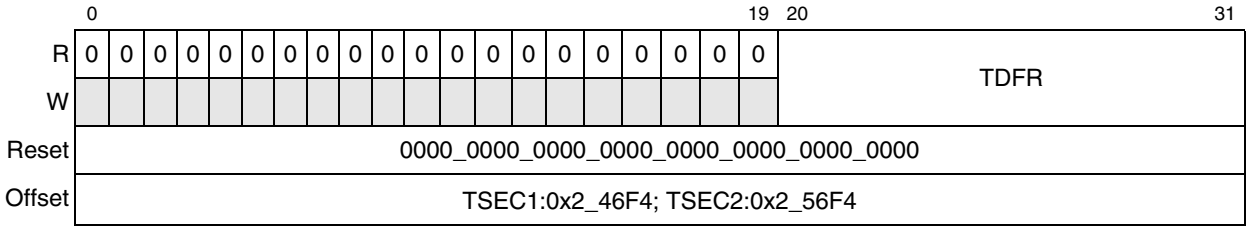


Figure 14-78. Transmit Deferral Packet Counter Register Definition

Table 14-76 describes the fields of the TDFR register.

Table 14-76. TDFR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TDFR	Transmit deferral packet counter. Increments for each frame, which was deferred on its first transmission attempt. This count does not include frames involved in collisions.

14.5.3.7.31 Transmit Excessive Deferral Packet Counter Register (TEDF)

Figure 14-79 shows the TEDF register.

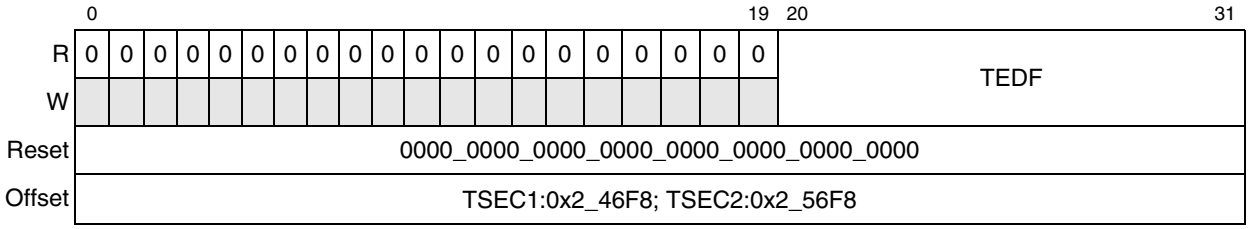


Figure 14-79. Transmit Excessive Deferral Packet Counter Register Definition

Table 14-77 describes the fields of the TEDF register.

Table 14-77. TEDF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TEDF	Transmit excessive deferral packet counter. Increments for frames aborted which were deferred for an excessive period of time (3036-byte times).

14.5.3.7.32 Transmit Single Collision Packet Counter Register (TSCL)

Figure 14-80 shows the TSCL register.

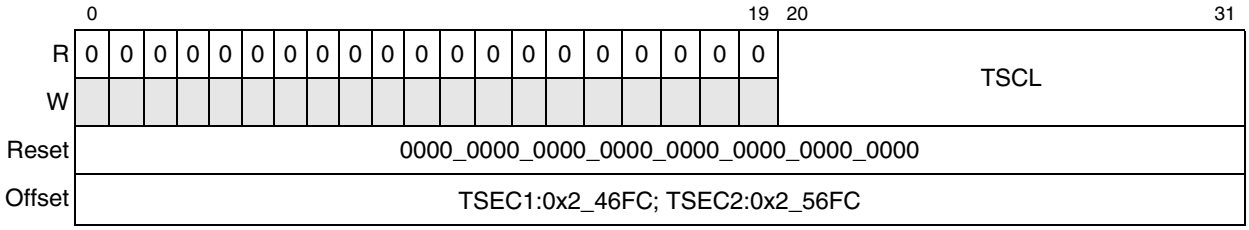


Figure 14-80. Transmit Single Collision Packet Counter Register Definition

Table 14-78 describes the fields of the TSCL register.

Table 14-78. TSCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TSCL	Transmit single collision packet counter. Increments for each frame transmitted which experienced exactly one collision during transmission.

14.5.3.7.33 Transmit Multiple Collision Packet Counter Register (TMCL)

Figure 14-81 shows the TMCL register.

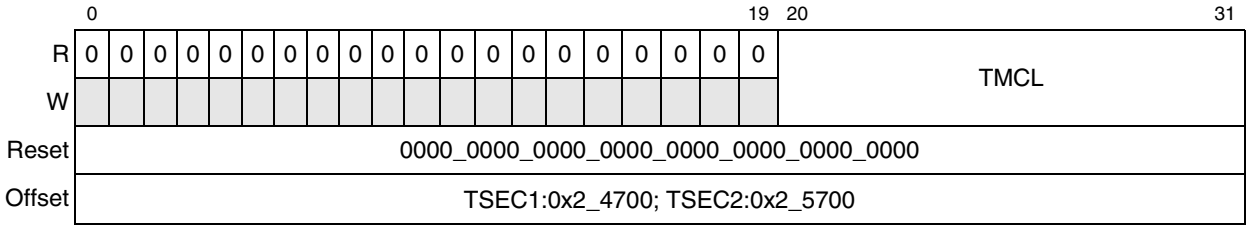


Figure 14-81. Transmit Multiple Collision Packet Counter Register Definition

Table 14-79 describes the fields of the TMCL register.

Table 14-79. TMCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TMCL	Transmit multiple collision packet counter. Increments for each frame transmitted which experienced 2-15 collisions (including any late collisions) during transmission as defined using the Half_Duplex[RETRANSMISSION MAXIMUM] field.

14.5.3.7.34 Transmit Late Collision Packet Counter Register (TLCL)

Figure 14-82 shows the TLCL register.

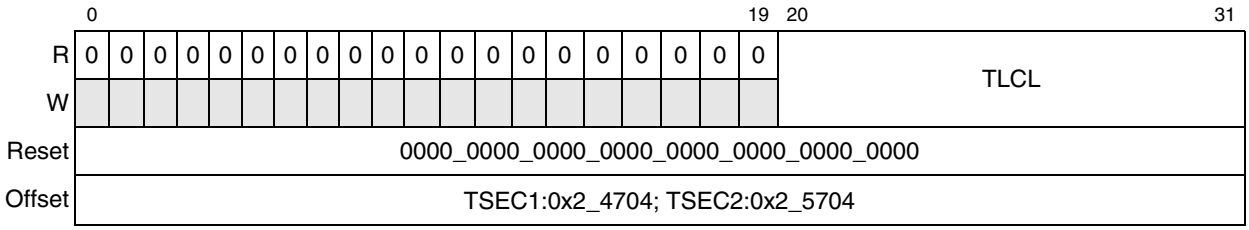


Figure 14-82. Transmit Late Collision Packet Counter Register Definition

Table 14-80 describes the fields of the TLCL register.

Table 14-80. TLCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TLCL	Transmit late collision packet counter. Increments for each frame transmitted which experienced a late collision during a transmission attempt.

14.5.3.7.35 Transmit Excessive Collision Packet Counter Register (TXCL)

Figure 14-83 shows the TXCL register.

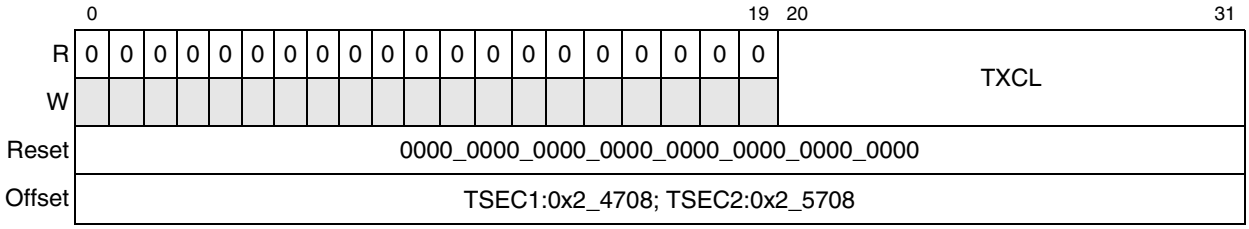


Figure 14-83. Transmit Excessive Collision Packet Counter Register Definition

Table 14-81 describes the fields of the TXCL register.

Table 14-81. TXCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCL	Transmit excessive collision packet counter. Increments for each frame that experienced 16 collisions during transmission and was aborted.

14.5.3.7.36 Transmit Total Collision Counter Register (TNCL)

Figure 14-84 shows the TNCL register.

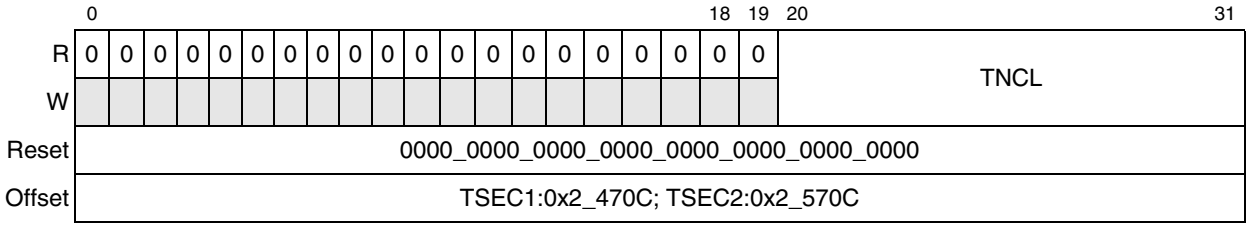


Figure 14-84. Transmit Total Collision Counter Register Definition

Table 14-82 describes the fields of the TNCL register.

Table 14-82. TNCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TNCL	Transmit total collision counter. Increments by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (that is, transmitting and receiving at the same time). **Note:** This count does not include collisions that result in an excessive collision condition.

14.5.3.7.37 Transmit Drop Frame Counter Register (TDRP)

Figure 14-85 shows the TDRP register.

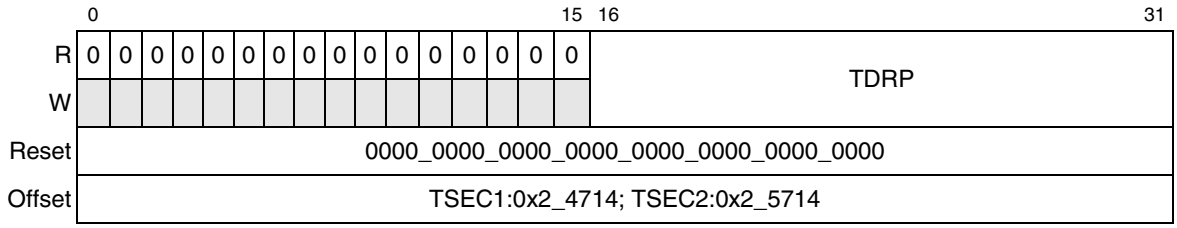


Figure 14-85. Transmit Drop Frame Counter Register Definition

Table 14-83 describes the fields of the TDRP register.

Table 14-83. TDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time a memory error or an underrun has occurred.

14.5.3.7.38 Transmit Jabber Frame Counter Register (TJBR)

Figure 14-86 shows the TJBR register.

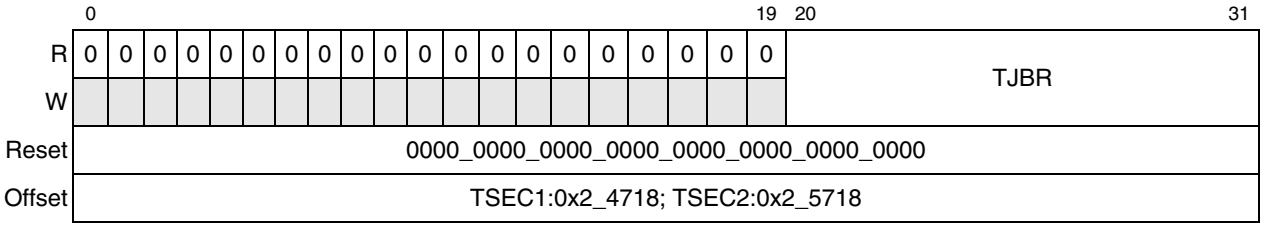


Figure 14-86. Transmit Jabber Frame Counter Register Definition

Table 14-84 describes the fields of the TJBR register.

Table 14-84. TJBR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TJBR	Transmit jabber frame counter. Increments for each oversized transmitted frame with an incorrect FCS value.

14.5.3.7.39 Transmit FCS Error Counter Register (TFCS)

Figure 14-87 shows the TFCS register.

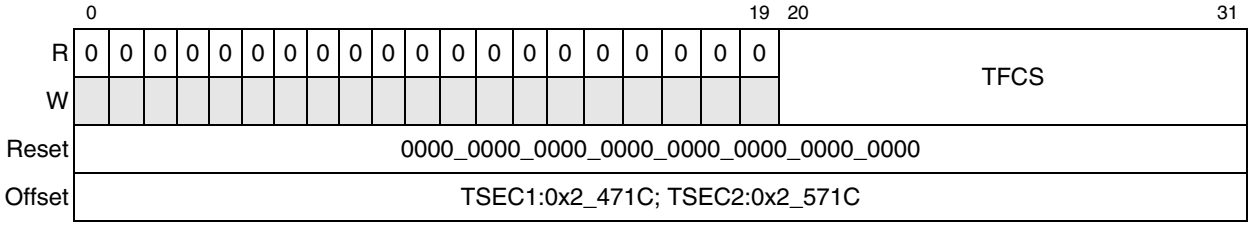


Figure 14-87. Transmit FCS Error Counter Register Definition

Table 14-85 describes the fields of the TFCS register.

Table 14-85. TFCS Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFCS	Transmit FCS error counter. Increments for every valid sized packet with an incorrect FCS value.

14.5.3.7.40 Transmit Control Frame Counter Register (TXCF)

Figure 14-88 shows the TXCF register.

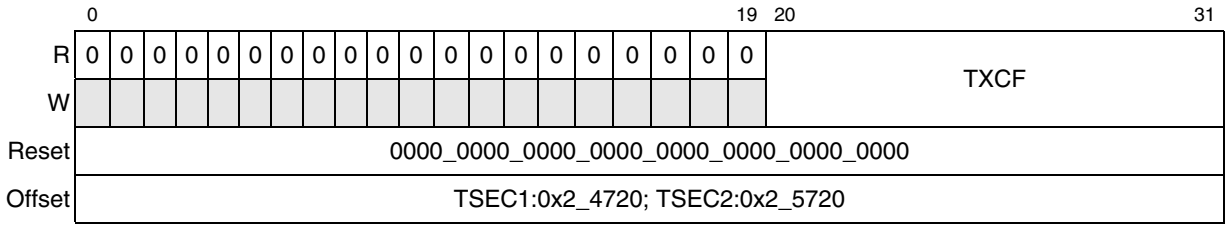


Figure 14-88. Transmit Control Frame Counter Register Definition

Table 14-86 describes the fields of the TXCF register.

Table 14-86. TXCF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCF	Transmit control frame counter. Increments for every valid size frame with a type field signifying a control frame.

14.5.3.7.41 Transmit Oversize Frame Counter Register (TOVR)

Figure 14-89 shows the TOVR register.

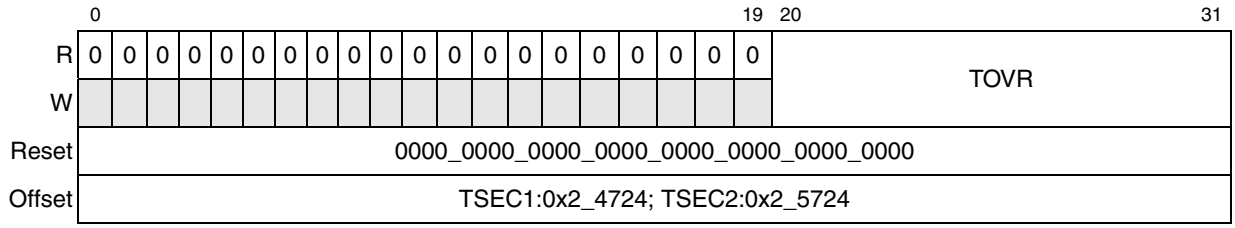


Figure 14-89. Transmit Oversized Frame Counter Register Definition

Table 14-87 describes the fields of the TOVR register.

Table 14-87. TOVR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TOVR	Transmit oversize frame counter. Increments for each oversized transmitted frame with a correct FCS value.

14.5.3.7.42 Transmit Undersize Frame Counter Register (TUND)

Figure 14-90 shows the TUND register.

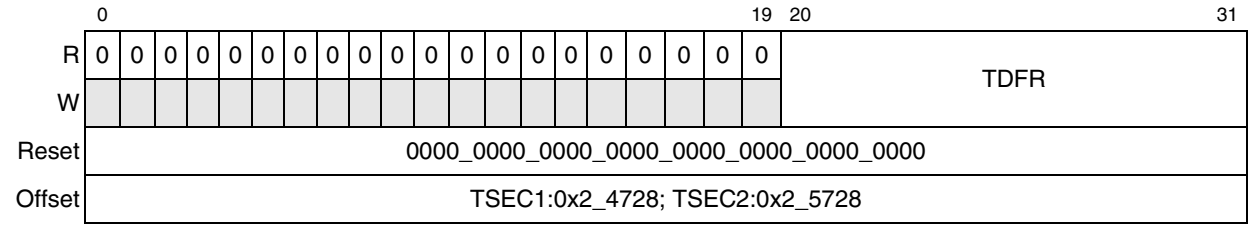


Figure 14-90. Transmit Undersize Frame Counter Register Definition

Table 14-88 describes the fields of the TUND register.

Table 14-88. TUND Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TDFR	Transmit undersize frame counter. Increments for every frame less then 64 bytes, with a correct FCS value.

14.5.3.7.43 Transmit Fragment Counter Register (TFRG)

Figure 14-91 shows the TFRG register.

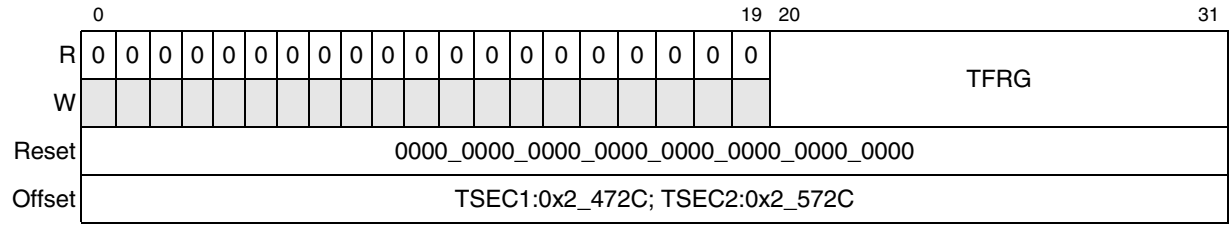


Figure 14-91. Transmit Fragment Counter Register Definition

Table 14-89 describes the fields of the TFRG register.

Table 14-89. TFRG Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFRG	Transmit fragment counter. Increments for every frame less then 64 bytes, with an incorrect FCS value.

14.5.3.7.44 Carry Register 1 (CAR1)

Carry register bits are cleared when written with a one. Figure 14-92 shows the CAR1 register.

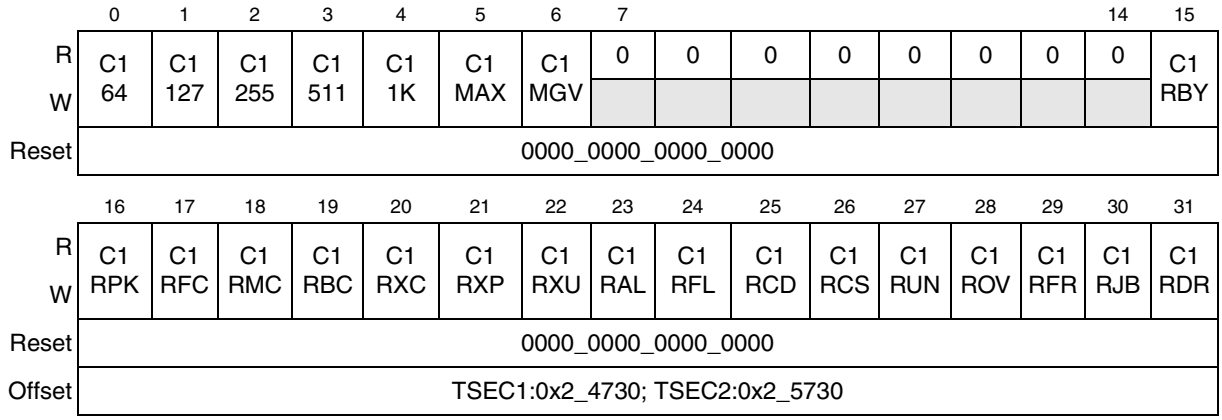


Figure 14-92. Carry Register 1 (CAR1) Register Definition

Table 14-90 describes the fields of the CAR1 register.

Table 14-90. CAR1 Field Descriptions

Bits	Name	Description
0	C164	Carry register 1 TR64 counter carry bit
1	C1127	Carry register 1 TR127 counter carry bit
2	C1255	Carry register 1 TR255 counter carry bit
3	C1511	Carry register 1 TR511 counter carry bit
4	C11K	Carry register 1 TR1K counter carry bit

Table 14-90. CAR1 Field Descriptions (continued)

Bits	Name	Description
5	C1MAX	Carry register 1 TRMAX counter carry bit
6	C1MGV	Carry register 1 TRMGV counter carry bit
7–14	—	Reserved
15	C1RBY	Carry register 1 RBYT counter carry bit
16	C1RPK	Carry register 1 RPKT counter carry bit
17	C1RFC	Carry register 1 RFCS counter carry bit
18	C1RMC	Carry register 1 RMCA counter carry bit
19	C1RBC	Carry register 1 RBCA counter carry bit
20	C1RXC	Carry register 1 RXCF counter carry bit
21	C1RXP	Carry register 1 RXPF counter carry bit
22	C1RXU	Carry register 1 RXUO counter carry bit
23	C1RAL	Carry register 1 RALN counter carry bit
24	C1RFL	Carry register 1 RFLR counter carry bit
25	C1RCD	Carry register 1 RCDE counter carry bit
26	C1RCS	Carry register 1 RCSE counter carry bit
27	C1RUN	Carry register 1 RUND counter carry bit
28	C1ROV	Carry register 1 ROVR counter carry bit
29	C1RFR	Carry register 1 RFRG counter carry bit
30	C1RJB	Carry register 1 RJBR counter carry bit
31	C1RDR	Carry register 1 RDRP counter carry bit

14.5.3.7.45 Carry Register 2 (CAR2)

Figure 14-93 shows the CAR2 register.

	0										11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	C2 TJB	C2 TFC	C2 TCF	C2 TOV
W															
Reset	0000_0000_0000_0000														

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2 TMC	C2 TBC	C2 TPF	C2 TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	0	C2 TDP
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4734; TSEC2:0x2_5734															

Figure 14-93. Carry Register 2 (CAR2) Register Definition

Carry register bits are cleared when written with a one. Table 14-91 describes the fields of the CAR2 register.

Table 14-91. CAR2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	C2TJB	Carry register 2 TJBR Counter Carry bit
13	C2TFC	Carry register 2 TFCS Counter Carry bit
14	C2TCF	Carry register 2 TXCF Counter Carry bit
15	C2TOV	Carry register 2 TOVR Counter Carry bit
16	C2TUN	Carry register 2 TUND Counter Carry bit
17	C2TFG	Carry register 2 TFRG Counter Carry bit
18	C2TBY	Carry register 2 TBYT Counter Carry bit
19	C2TPK	Carry register 2 TPKT Counter Carry bit
20	C2TMC	Carry register 2 TMCA Counter Carry bit
21	C2TBC	Carry register 2 TBCA Counter Carry bit
22	C2TPF	Carry register 2 TXPF Counter Carry bit
23	C2TDF	Carry register 2 TDFR Counter Carry bit
24	C2TED	Carry register 2 TEDF Counter Carry bit
25	C2TSC	Carry register 2 TSCL Counter Carry bit
26	C2TMA	Carry register 2 TMCL Counter Carry bit
27	C2TLC	Carry register 2 TLCL Counter Carry bit
28	C2TXC	Carry register 2 TXCL Counter Carry bit
29	C2TNC	Carry register 2 TNCL Counter Carry bit
30	—	Reserved
31	C2TDP	Carry register 2 TDRP Counter Carry bit

14.5.3.7.46 Carry Mask Register 1 (CAM1)

As long as one of the below mask bits is cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits all default to a set state. Figure 14-94 shows the CAM1 register.

	0	1	2	3	4	5	6	7							14	15
R	M1 64	M1 127	M1 255	M1 511	M1 1K	M1 MAX	M1 MGV	0	0	0	0	0	0	0	0	M1 RBY
W																
Reset	1111_1110_0000_0001															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M1 RPK	M1 RFC	M1 RMC	M1 RBC	M1 RXC	M1 RXP	M1 RXU	M1 RAL	M1 RFL	M1 RCD	M1 RCS	M1 RUN	M1 ROV	M1 RFR	M1 RJB	M1 RDR
W																
Reset	1111_1111_1111_1111															
Offset	TSEC1:0x2_4738; TSEC2:0x2_5738															

Figure 14-94. Carry Mask Register 1 (CAM1) Register Definition

Table 14-92 describes the fields of the CAM1 register.

Table 14-92. CAM1 Field Descriptions

Bits	Name	Description
0	M164	Mask register 1 TR64 counter carry bit mask
1	M1127	Mask register 1 TR127 counter carry bit mask
2	M1255	Mask register 1 TR255 counter carry bit mask
3	M1511	Mask register 1 TR511 counter carry bit mask
4	M11K	Mask register 1 TR1K counter carry bit mask
5	M1MAX	Mask register 1 TRMAX counter carry bit mask
6	M1MGV	Mask register 1 TRMGV counter carry bit mask
7–14	—	Reserved
15	M1RBY	Mask register 1 RBYT counter carry bit mask
16	M1RPK	Mask register 1 RPKT counter carry bit mask
17	M1RFC	Mask register 1 RFCS counter carry bit mask
18	M1RMC	Mask register 1 RMCA counter carry bit mask
19	M1RBC	Mask register 1 RBCA counter carry bit mask
20	M1RXC	Mask register 1 RXCF counter carry bit mask
21	M1RXP	Mask register 1 RXPF counter carry bit mask
22	M1RXU	Mask register 1 RXUO counter carry bit mask
23	M1RAL	Mask register 1 RALN counter carry bit mask
24	M1RFL	Mask register 1 RFLR counter carry bit mask
25	M1RCD	Mask register 1 RCDE counter carry bit mask

Table 14-92. CAM1 Field Descriptions (continued)

Bits	Name	Description
26	M1RCS	Mask register 1 RCSE counter carry bit mask
27	M1RUN	Mask register 1 RUND counter carry bit mask
28	M1ROV	Mask register 1 ROVR counter carry bit mask
29	M1RFR	Mask register 1 RFRG counter carry bit mask
30	M1RJB	Mask register 1 RJBR counter carry bit mask
31	M1RDR	Mask register 1 RDRP counter carry bit mask

14.5.3.7.47 Carry Mask Register 2 (CAM2)

As long as one of the below mask bits is cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits default to a set state. Figure 14-95 shows the CAM2 register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	M2 TJB	M2 TFC	M2 TCF	M2 TOV
W																
Reset	0000_0000_0000_1111															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2 TBC	M2 TPF	M2 TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	0	M2 TDP
W																
Reset	1111_1111_1111_1111															
Offset	TSEC1:0x2_473C; TSEC2:0x2_573C															

Figure 14-95. Carry Mask Register 2 (CAM2) Register Definition

Table 14-93 describes the fields of the CAM2 register.

Table 14-93. CAM2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	M2TJB	Mask register 2 TJBR counter carry bit mask
13	M2TFC	Mask register 2 TFCS counter carry bit mask
14	M2TCF	Mask register 2 TXCF counter carry bit mask
15	M2TOV	Mask register 2 TOVR counter carry bit mask
16	M2TUN	Mask register 2 TUND counter carry bit mask
17	M2TFG	Mask register 2 TFRG counter carry bit mask
18	M2TBY	Mask register 2 TBYT counter carry bit mask
19	M2TPK	Mask register 2 TPKT counter carry bit mask
20	M2TMC	Mask register 2 TMCA counter carry bit mask

Table 14-93. CAM2 Field Descriptions (continued)

Bits	Name	Description
21	M2TBC	Mask register 2 TBCA counter carry bit mask
22	M2TPF	Mask register 2 TXPF counter carry bit mask
23	M2TDF	Mask register 2 TDFR counter carry bit mask
24	M2TED	Mask register 2 TEDF counter carry bit mask
25	M2TSC	Mask register 2 TSCL counter carry bit mask
26	M2TMA	Mask register 2 TMCL counter carry bit mask
27	M2TLC	Mask register 2 TLCL counter carry bit mask
28	M2TXC	Mask register 2 TXCL counter carry bit mask
29	M2TNC	Mask register 2 TNCL counter carry bit mask
30	—	Reserved
31	M2TDP	Mask register 2 TDRP counter carry bit mask

14.5.3.8 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. If the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to a hash table entry. The user can enable a hash entry by setting the appropriate bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. The user must further filter the address. See Section 14.6.2.6.2, "Hash Table Algorithm," for more information on the hash algorithm.

14.5.3.8.1 Individual Address Registers 0–7 (IADDR*n*)

The IADDR*n* registers, shown in Figure 14-96, are written by the user. These registers represent 256 entries of the individual (unicast) address hash table used in the address recognition process. When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 high-order bits (0-7) of the CRC remainder are mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry.

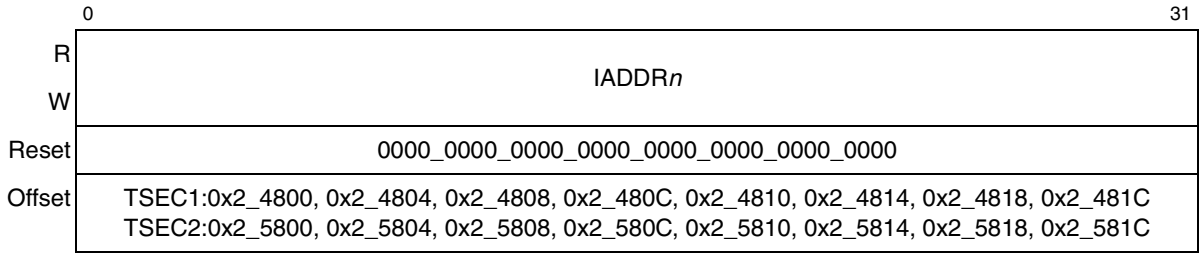


Figure 14-96. IADDR*n* Register Definition

Table 14-94 describes the field of the IADDR*n* registers.

Table 14-94. IADDR*n* Field Descriptions

Bits	Name	Description
0–31	IADDR*n*	Represents the 32-bit value associated with the corresponding register. IADDR0 contains the high-order 32 bits of the 256-entry hash table and IADDR7 represents the low-order 32 bits. For instance, the MSB of IADDR0 correlates to entry 0 and the LSB of IADDR7 correlates to entry 255.

14.5.3.8.2 Group Address Registers 0–7 (GADDR*n*)

The GADDR*n* registers are written by the user. Together these registers represent 256 entries of the group (multicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Figure 14-97 shows the GADDR*n* registers.

	0	31
R W	GADDR*n*	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	TSEC1:0x2_4880, 0x2_4884, 0x2_4888, 0x2_488C, 0x2_4890, 0x2_4894, 0x2_4898, 0x2_489C TSEC2:0x2_5880, 0x2_5884, 0x2_5888, 0x2_588C, 0x2_5890, 0x2_5894, 0x2_5898, 0x2_589C	

Figure 14-97. GADDR*n* Register Definition

Table 14-95 describes GADDR*n*.

Table 14-95. GADDR*n* Field Descriptions

Bits	Name	Description
0–31	GADDR*n*	Represents the 32-bit value associated with the corresponding register. GADDR0 contains the high-order 32 bits of the 256 entry group hash table and GADDR7 represents the low-order 32 bits. For instance, the MSB of GADDR0 correlates to entry 0 and the LSB of GADDR7 correlates to entry 255.

14.5.3.9 Attribute Registers

This section describes the two TSEC frame attribute registers.

14.5.3.9.1 Attribute Register (ATTR)

The attribute register, shown in Figure 14-98, defines attributes and transaction types used to access buffer descriptors, to write receive data, and to read transmit data. Snoop enable attributes may be set for reading buffer descriptors and for reading transmit data. Buffer descriptors may be written with attributes that cause allocation into the L2 cache with or without line locking. Similarly, sections of a receive frame header may have attributes attached that cause allocation and locking in the L2 cache. This process of

specifying a region of each frame to stash into the L2 cache is referred to as extraction, which is specified in conjunction with ATTRELI. ATTR[ELCWT] only has meaning if ATTRELI[EL] is non-zero.

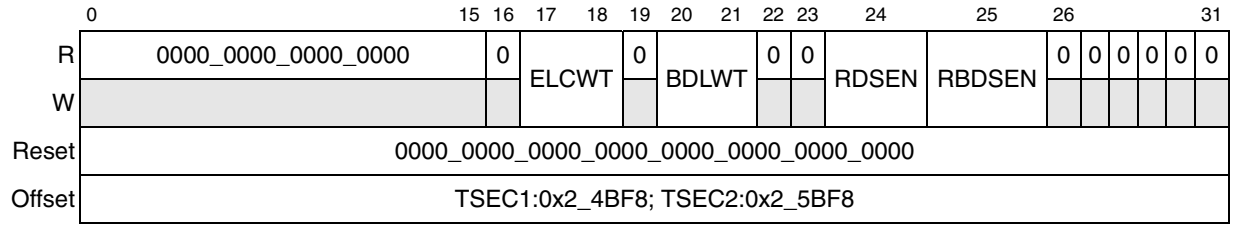


Figure 14-98. ATTR Register Definition

Table 14-96 describes the fields of the ATTR register.

Table 14-96. ATTR Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17–18	ELCWT	Extracted L2 cache write type. Specifies the write transaction type to perform for the extracted data. This occurs only if ATTRELI[EL] is non-zero. For maximum performance, it is recommended that if ELCWT is set to allocate, BDLWT must also be set to allocate. Writes to cache are always performed with snoop. This setting overrides the RDSEN bit setting. 00 No allocation performed. 01 Reserved, no extraction occurs. 10 Allocate L2 cache line. 11 Reserved
19	—	Reserved
20–21	BDLWT	Buffer descriptor L2 cache write type. Specifies the write transaction type to perform for the buffer descriptor for a receive frame. Writes to cache are always performed with snoop. 00 No allocation performed. This setting overrides the RBDSEN bit setting. 01 Reserved 10 Allocate L2 cache line. 11 Reserved
22–23	—	Reserved
24	RDSEN	Rx data snoop enable. This bit is superseded by the ELCWT settings. 0 Disables snooping of all receive frames data to memory unless ELCWT specifies L2 allocation. 1 Enables snooping of all receive frames data to memory.
25	RBDSEN	RxBD snoop enable. This bit is superseded by the BDLWT settings. 0 Disables snooping of all receive BD memory accesses unless BDLWT specifies L2 allocation. 1 Enables snooping of all receive BD memory accesses.
26–31	—	Reserved

14.5.3.9.2 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the extract index and extract length. Figure 14-99 shows the ATTRELI register.

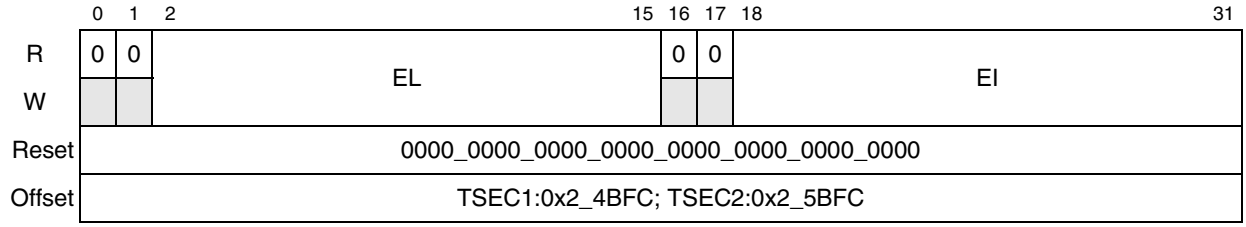


Figure 14-99. ATTRELI Register Definition

Table 14-97 describes the fields of the ATTRELI register.

Table 14-97. ATTRELI Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–15	EL	Extracted length. Specifies the number of bytes to extract from the receive frame. The DMA controller uses this field to perform extraction. If set to zero, no extraction is performed.
16–17	—	Reserved
18–31	EI	Extracted index. Points to the first byte within the receive frame from which to begin extracting data.

14.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

14.5.4.1 TBI MII Set Register Descriptions

This section describes the TBI MII registers. All TBI registers are 16 bits wide and are accessed at the offset of the TBI physical address. The TBI physical address of the TSEC is stored in TBIPA. Writing to the TBI registers is similar to writing to an external PHY, by using the MII management interface. By using TBIPA in place of the PHY address, in the MIIMADD[PHY Address] field, and setting MIIMADD[Register Address] to the address offset that corresponds to the desired register (see Table 14-98), the user can read (set MIIMCOM[Read Cycle]) or write (writing to MIIMCON[PHY control]) to the TBI block. Refer to the TBI physical address register in Section 14.5.3.1, “TSEC General Control and Status Registers,” and the TBI MII register set in Table 14-98.

Note that jitter diagnostics and TBI control are not IEEE 802.3 required registers and are only used for test and control of the TSEC TBI block. The TBI’s TBI control register (TBI) is for configuring the TSEC

ten-bit interface block. However, because this TBI block has an MII management interface (just like any other PHY), it has an IEEE 802.3 register called the control register (CR).

Table 14-98. TBI MII Register Set

Offset	Name	Access	Size	Section/Page
TEN-BIT INTERFACE (TBI) REGISTERS				14.5.4/14-87
0x00	Control (CR)	R/W[1]	16 bits	14.5.4.2/14-88
0x01	Status (SR)	R, LH, LL	16 bits	14.5.4.3/14-89
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	14.5.4.3/14-89
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	14.5.4.5/14-92
0x06	AN expansion (ANEX)	R, LH	16 bits	14.5.4.6/14-93
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	14.5.4.7/14-94
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	14.5.4.8/14-95
0x0F	Extended status (EXST)	R	16 bits	14.5.4.9/14-96
0x10	Jitter diagnostics (JD)	R/W	16 bits	14.5.4.10/14-97
0x11	TBI control (TBICON)	R/W	16 bits	14.5.4.11/14-98

[1] R = means read only, WO = write only, R/W = read and write, LH = latches high, LL = latches low, SC = self-clearing,

14.5.4.2 Control Register (CR)

Figure 14-100 shows the CR register.

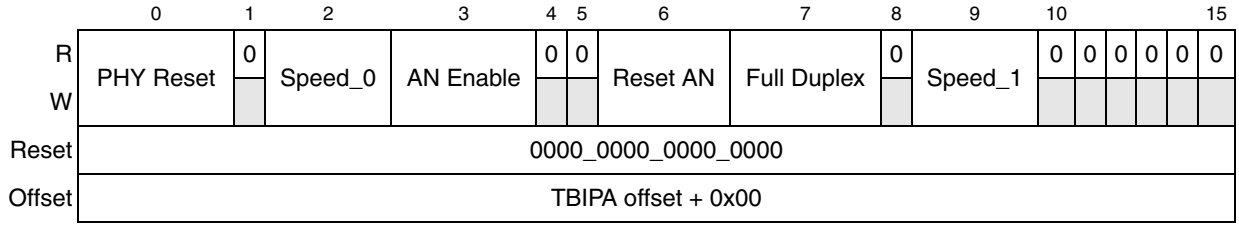


Figure 14-100. Control Register Definition

Table 14-99 describes the fields of the CR register.

Table 14-99. CR Field Descriptions

Bits	Name	Description
0	PHY Reset	PHY reset. This bit is cleared by default. This bit is self-clearing. 0 Normal operation. 1 The control and status registers are returned to their default value. This in turn may change the internal state of the TBI and its link partner.
1	—	Reserved

Table 14-99. CR Field Descriptions (continued)

Bits	Name	Description
2	Speed_0	Speed selection. This bit defaults to a cleared state and must always be cleared, which corresponds to 1-Gbps speed.
3	AN Enable	Auto-negotiation enable. This bit is cleared by default. 0 The values programmed in bits 2, 7 and 9 determine the operating condition of the link. 1 Auto-negotiation process enabled.
4–5	—	Reserved
6	Reset AN	Reset auto-negotiation. This bit is cleared by default and is self-clearing. 0 Normal operation. 1 The auto-negotiation process restarts. This action is only available if auto-negotiation is enabled.
7	Full Duplex	Duplex mode. This bit is cleared by default. 0 Half-duplex operation. 1 Full-duplex operation.
8	—	Reserved, must be cleared.
9	Speed_1	Speed selection. Defaults to a cleared state but must always be set, which corresponds to 1-Gbps.
10–15	—	Reserved

Maximum Operating Speed	Bit 2	Bit 9
Reserved	0	0
Reserved	1	0
1 Gbps	0	1
Reserved	1	1

14.5.4.3 Status Register (SR)

Figure 14-101 shows the SR register.

	0						6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	Extend Status	0	No Pre	AN Done	Remote Fault	AN Ability	Link Status	0	Extend Ability
W																
Reset	0000_0001_0100_1001															
Offset	TBIPA offset + 0x01															

Figure 14-101. Status Register Definition

Table 14-100 describes the fields of the SR register.

Table 14-100. SR Descriptions

Bits	Name	Description
0–6	—	Reserved, must be cleared.
7	Extend Status	Indicates that PHY status information is also contained in the Extended Status Register. Returns 1 on read. This bit is read-only.
8	—	Reserved, must be cleared.
9	No Pre	MF preamble suppression enable. This bit indicates whether or not the PHY is capable of handling MII management frames without the 32-bit preamble field. Returns 1, indicating support for suppressed preamble MII management frames. This bit is read-only.
10	AN Done	Auto-negotiation complete. This bit is read-only and is cleared by default. 0 Either the auto-negotiation process is underway or the auto-negotiation function is disabled. 1 The auto-negotiation process has completed.
11	Remote Fault	Remote fault. This bit is read-only and is cleared by default. Each read of SR clears this bit. 0 Normal operation. 1 A remote fault condition was detected. Latches high for software to detect the condition.
12	AN Ability	Auto-negotiation ability. While read as set, this bit indicates that the PHY has the ability to perform auto-negotiation. While read as cleared, this bit indicates the PHY lacks the ability to perform auto-negotiation. Returns 1 on read. This bit is read-only.
13	Link Status	Link status. This bit is read-only and is cleared by default. 0 A valid link is not established. Latches low allowing for software polling to detect a failure condition. 1 A valid link is established.
14	—	Reserved, must be cleared.
15	Extend Ability	Extended capability. This bit indicates that the PHY contains the extended set of registers (those beyond control and status). Returns 1 on read. This bit is read-only.

14.5.4.4 AN Advertisement Register (ANA)

Figure 14-102 shows the ANA register.

	0	1	2 3	4		6	7 8	9	10	11				15
R	Next Page	Ack	Remote Fault	0	0	0	Pause	Half Duplex	Full Duplex	0	0	0	0	0
W														
Reset	0000_0000_0000_0000													
Offset	TBIPA offset + 0x04													

Figure 14-102. AN Advertisement Register Definition

Table 14-101 describes the fields of the ANA register.

Table 14-101. ANA Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Next Page</td><td>Next page configuration. The local device sets this bit to either request next page transmission or advertise next page exchange capability.
0 The local device wishes not to engage in next page exchange.
1 The local device has no next pages but wishes to allow reception of next pages. If the local device has no next pages and the link partner wishes to send next pages, the local device shall send null message codes and have the message page set to 0b000_0000_0001, as defined in annex 28C of the IEEE 802.3 specification.</td></tr>
<tr><td>1</td><td>Ack</td><td>Acknowledge. Write 0, ignore on read.This bit is read-only. (Reserved)</td></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The local device’s remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the following table. The default value is 00. Indicate a fault by setting a non-zero remote fault encoding and re-negotiating.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, must be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>The local device’s PAUSE capability is encoded in bits 7 and 8, and the decodes are shown in the following table. For priority resolution information consult Table 14-102.
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability
0 Designates local device as not capable of half-duplex operation.
1 Designates local device as capable of half-duplex operation.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability
0 Designates the local device as not capable of full-duplex operation.
1 Designates the local device as capable of full-duplex operation.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, must be cleared.</td></tr>
</table>

Table 14-102 describes the resolution of pause priority.

Table 14-102. PAUSE Priority Resolution

Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	0	x	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	1	Enable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Enable PAUSE receive
1	0	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	0	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive
1	1	0	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	1	0	1	Disable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Disable PAUSE receive
1	1	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive

14.5.4.5 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 14-103 shows the ANLPBPA register.

	0	1	2	3	4		6	7	8	9	10	11				15
R	Next Page	Ack	Remote Fault		0	0	0	Pause		Half Duplex	Full Duplex	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TBIPA offset + 0x05															

Figure 14-103. AN Link Partner Base Page Ability Register Definition

Table 14-103 describes the fields of the ANLPBPA register.

Table 14-103. ANLPBPA Field Descriptions

Bits	Name	Description
0	Next Page	Next page. This bit is read-only. The link partner sets or clears this bit. 0 Link partner has no subsequent next pages or is not capable of receiving next pages. 1 Link partner either requesting next page transmission or indicating the capability to receive next pages.
1	Ack	Acknowledge. Ignore on read. This bit is read-only

Table 14-103. ANLPBPA Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The link partner's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the remote fault encoding field table below. This bit is read-only.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, must be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>Encoding of the link partner's PAUSE capability is shown in the PAUSE encoding table below. For priority resolution information consult the IEEE 802.3 specification. This bit is read-only.
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability. This bit is read-only.
0 Link partner is not capable of half-duplex mode.
1 Link partner is capable of half-duplex mode.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability. This bit is read-only.
0 Link partner is not capable of full-duplex mode.
1 Link partner is capable of full-duplex mode.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, must be cleared.</td></tr>
</table>

14.5.4.6 AN Expansion Register (ANEX)

Figure 14-104 shows the ANEX register.

	0												12	13		14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	NP Able		Page Rx'd	0
W																	
Reset	0000_0000_0000_0100																
Offset	TBIPA offset + 0x06																

Figure 14-104. AN Expansion Register Definition

Table 14-104 describes the fields of the ANEX register.

Table 14-104. ANEX Field Descriptions

Bits	Name	Description
0–12	—	Reserved, must be cleared.
13	NP Able	Next page able. This bit is read-only and returns 1 on read. While read as set, indicates local device supports next page function.
14	Page Rx'd	Page received. This bit is read-only. The bit clears on a read to the register. 0 Normal operation. 1 A new page was received and stored in the applicable AN link partner ability or AN next page register. This bit latches high in order for software to detect while polling.
15	—	Reserved, must be cleared.

14.5.4.7 AN Next Page Transmit Register (ANNPT)

Figure 14-105 shows the ANNPT register.

	0	1	2	3	4	5 … 15
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message/Unformatted Code Field
W						
Reset	0000_0000_0000_0000					
Offset	TBIPA offset + 0x07					

Figure 14-105. AN Next Page Transmit Register Definition

Table 14-105 describes the fields of the ANNPT register.

Table 14-105. ANNPT Field Descriptions

Bits	Name	Description
0	Next Page	Next page indication. [Reference MII bit 7.15 in IEEE 802.3, 2000 Edition Clause 28.2.4] 0 Last page 1 Additional next pages to follow
1	Ack	Acknowledge. Write 0, ignore on read. [Reference MII bit 7.14] This bit is read-only.
2	Msg Page	Message page. [Reference MII bit 7.13] 0 Unformatted page 1 Message page
3	Ack2	Acknowledge 2. Used by the next page function to indicate that the device has the ability to comply with the message. [Reference MII bit 7.12] 0 The local device cannot comply with message. 1 The local device complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. [Reference MII bit 7.11] This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was set. 1 Toggle bit of the previously-exchanged link code word was clear.
5–15	Message/ Unformatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value. [Reference MII field 7.10:0]

14.5.4.8 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 14-106 shows the ANLPANP register.

	0	1	2	3	4	5										15
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message/Unformatted Code Field										
W																
Reset	0000_0000_0000_0000															
Offset	TBIPA offset + 0x08															

Figure 14-106. AN Link Partner Ability Next Page Register Definition

Table 14-106 describes the ANLPANP fields.

Table 14-106. ANLPANP Field Descriptions

Bits	Name	Description
0	Next Page	Next page. The link partner sets and clears this bit. 0 Last page from link partner. 1 Additional next pages to follow.
1	Ack	Acknowledge. Ignore on read. [Reference MII bit 8.14] This bit is read-only.
2	Msg Page	Message page 0 unformatted page. 1 message page.
3	Ack2	Acknowledge 2. Indicates the link partner's ability to comply with the message. 0 Link partner cannot comply with message. 1 Link partner complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was set. 1 Toggle bit of the previously-exchanged link code word was clear.
5–15	Message/ Unformatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value.

14.5.4.9 Extended Status Register (EXST)

Figure 14-107 shows the EXST register.

	0	1	2	3	4											15
R	1000X Full	1000X Half	1000T Full	1000T Half	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1010_0000_0000_0000															
Offset	TBIPA offset + 0x0F															

Figure 14-107. Extended Status Register Definition

Table 14-107 describes the fields of the EXST register.

Table 14-107. EXST Field Descriptions

Bits	Name	Description
0	1000X Full	1000X full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operate in 1000BASE-X full-duplex mode. 1 PHY can operate in 1000BASE-X full-duplex mode.
1	1000X Half	1000X half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operate in 1000BASE-X half-duplex mode. 1 PHY can operate in 1000BASE-X half-duplex mode.
2	1000T Full	1000T full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operate in 1000BASE-T full-duplex mode. 1 PHY can operate in 1000BASE-T full-duplex mode.
3	1000T Half	1000T half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operate in 1000BASE-T half-duplex mode. 1 PHY can operate in 1000BASE-T half-duplex mode.
4–15	—	Reserved

14.5.4.10 Jitter Diagnostics Register (JD)

Annex 36A in IEEE 802.3z describes several jitter test patterns. These can be configured to be sent by writing the jitter diagnostics register. See the register description for more information. It may be wise to auto-negotiate and advertise a remote fault, signaling offline, prior to beginning the test patterns. Figure 14-108 shows the JD register.

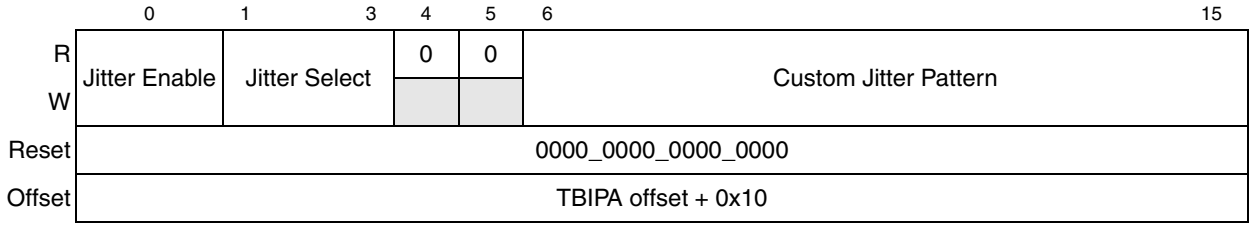


Figure 14-108. Jitter Diagnostics Register Definition

Table 14-108 describes the fields of the JD register.

Table 14-108. JD Field Descriptions

Bits	Name	Description
0	Jitter Enable	Jitter enable. This bit is cleared by default. 0 Normal transmit operation. 1 Enable the TBI to transmit the jitter test patterns defined in annex 36A of the IEEE 802.3 specification.
1–3	Jitter Select	Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 0.

Jitter Pattern Select	bit[1]	bit[2]	bit[3]
User defined uses custom jitter pattern, bits 6–15	0	0	0
High frequency (±D21.5) 1010101010101010101010101010101010101010…	0	0	1
Mixed frequency (±K28.5) 1111101011000001010011111010110000010100…	0	1	0
Low frequency 1111100000111110000011111000001111100000…	0	1	1
Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)	1	0	0
Square Wave (-K28.7) 0011111000001111100000111110000011111000…	1	0	1
Reserved	1	1	0
Reserved	1	1	1

4–5	—	Reserved	
6–15	Custom Jitter Pattern	Used in conjunction with jitter (pattern) select and jitter (diagnostic) enable; set this field to the desired custom pattern which is continuously transmitted. Its default is 0.	

14.5.4.11 TBI Control Register (TBICON)

Figure 14-109 shows the TBICON register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Soft_Reset	0	Disable Rx Dis	Disable Tx Dis	0	0	0	AN Sense	0	0	Clock Select	MII Mode	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TBIPA offset + 0x11															

Figure 14-109. TBI Control Register Definition

Table 14-109 describes the fields of the TBICON register.

Table 14-109. TBICON Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation. 1 Resets the functional modules in theTBI.
1	—	Reserved. (Ignore on read)
2	Disable Rx Dis	Disable receive disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the receive direction.
3	Disable Tx Dis	Disable transmit disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the transmit direction.
4–6	—	Reserved
7	AN Sense	Auto-negotiation sense enable. This bit is cleared by default. 0 IEEE 802.3z Clause 37 behavior is desired, which results in the link not completing. 1 Allow the auto-negotiation function to sense either a Gigabit MAC in auto-negotiation bypass mode or an older Gigabit MAC without auto-negotiation capability. If sensed, auto-negotiation complete becomes true; however, the page received is low, indicating no page was exchanged. Management can then act accordingly.
8–9	—	Reserved
10	Clock Select	Clock select. This bit is cleared by default. 0 Allow the TBI to accept dual split-phase 62.5-MHz receive clocks. 1 Configure the TBI to accept a 125-MHz receive clock from the SerDes/PHY. The 125-MHz clock must be physically connected to ‘PMA receive clock 0’.
11	MII Mode	This bit describes the configuration mode of the TBI. The user reads a 1 while the TBI is configured in GMII/MII mode (connected to a GMII/MII PHY) and a 0 while configured in TBI mode (connected to a 1000BASE-X SerDes). Its value is the inverse of ECNTRL[TBIM]. 0 TBI mode. 1 GMII mode.
12–13	—	Reserved
14–15	—	Reserved

14.6 Functional Description

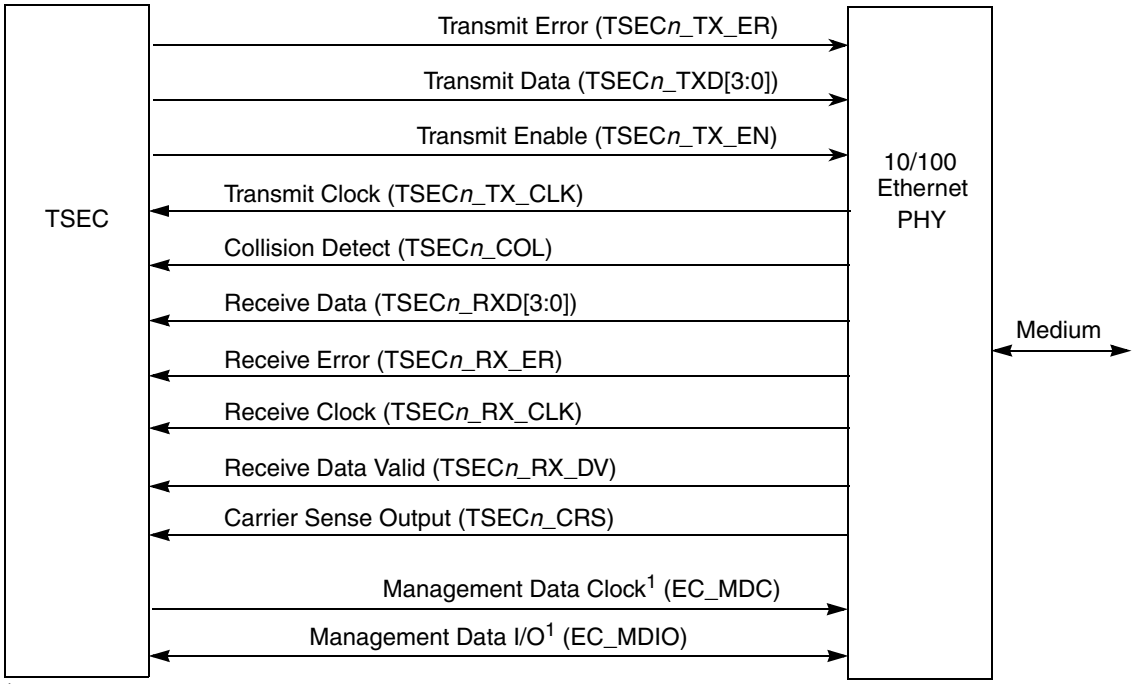
This section describes many of the functions of the TSEC controller.

14.6.1 Connecting to Physical Interfaces

This section describes how to connect the TSEC to various interfaces: MII, GMII, RGMII, RTBI, and TBI. To avoid confusion, all of the buses follow the bus conventions used in the IEEE 802.3 specification, because each PHY follows the same convention. (For instance, in the bus TSEC*n*_TXD[7:0], bit 7 is the MSB and bit 0 is the LSB.)

14.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the TSEC. Figure 14-110 shows the basic components of the MII including the signals required to establish TSEC module connection with a PHY.



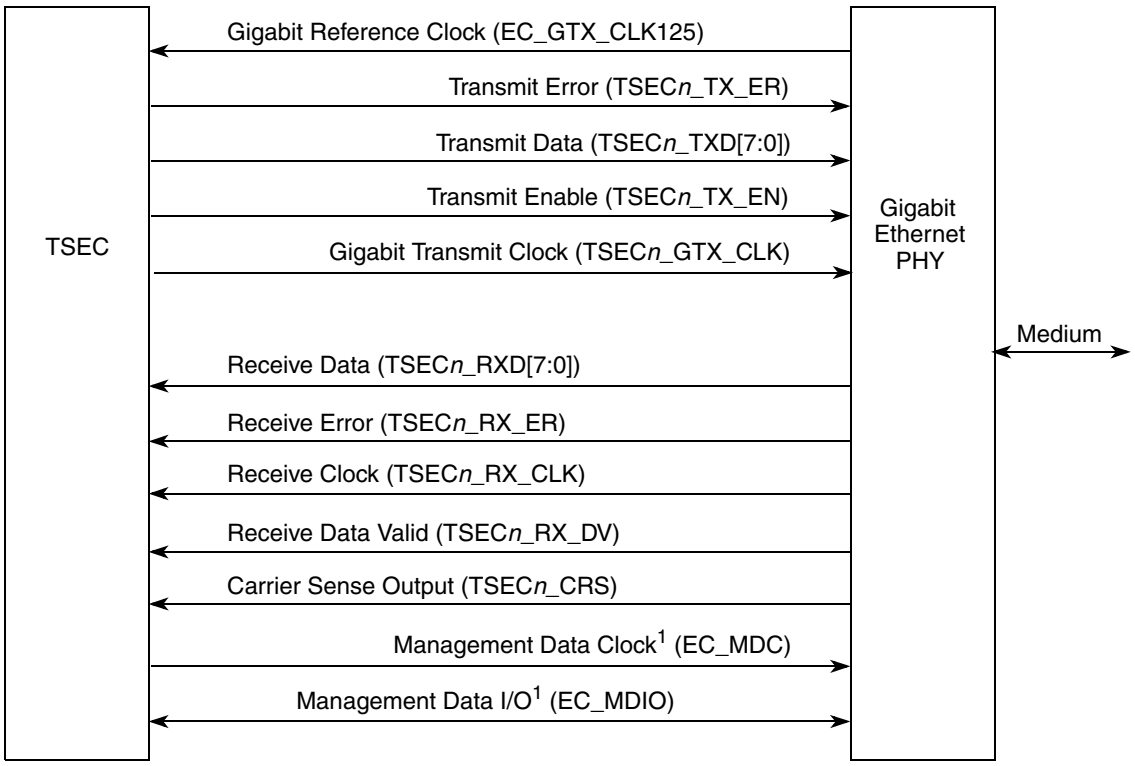
1 The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-110. TSEC-MII Connection

An MII interface has 18 signals (including the EC_MDC and EC_MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

14.6.1.2 Gigabit Media-Independent Interface (GMII)

This section describes the gigabit media-independent interface (GMII) intended to be used between the PHYs and the TSEC. Figure 14-111 shows the basic components of the GMII including the signals required to establish the TSEC module connection with a PHY.



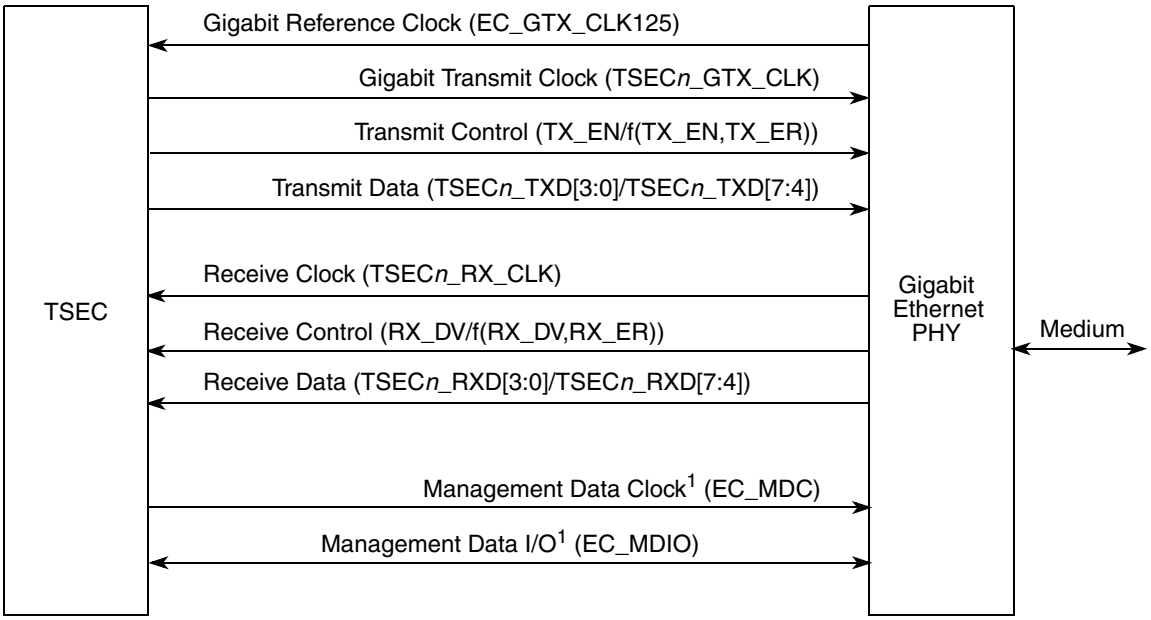
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-111. TSEC-GMII Connection

A GMII interface has 28 signals (TSEC*n*_GTX_CLK + EC_GTX_CLK125 included), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

14.6.1.3 Reduced Gigabit Media-Independent Interface (RGMII)

This section describes the reduced gigabit media-independent interface (RGMII) intended to be used between the PHYs and the GMII MAC. The RGMII is an alternative to the IEEE802.3u MII, the IEEE802.3z GMII, and the TBI. The RGMII reduces the number of signals required to interconnect the MAC and the PHY from a maximum of 28 signals (GMII) to 15 signals (EC_GTX_CLK125 included) in a cost-effective and technology-independent manner. To accomplish this objective, the data paths and all associated control signals are multiplexed using both edges of the clock. For gigabit operation, the clocks operate at 125 MHz, and for 10/100 operation, the clocks operate at 2.5 or 25 MHz, respectively. Figure 14-112 shows the basic components of the gigabit reduced media-independent interface and the signals required to establish the gigabit Ethernet controllers' module connection with a PHY. The RGMII is implemented as defined by the RGMII Specification Version 1.2a 9/22/00.

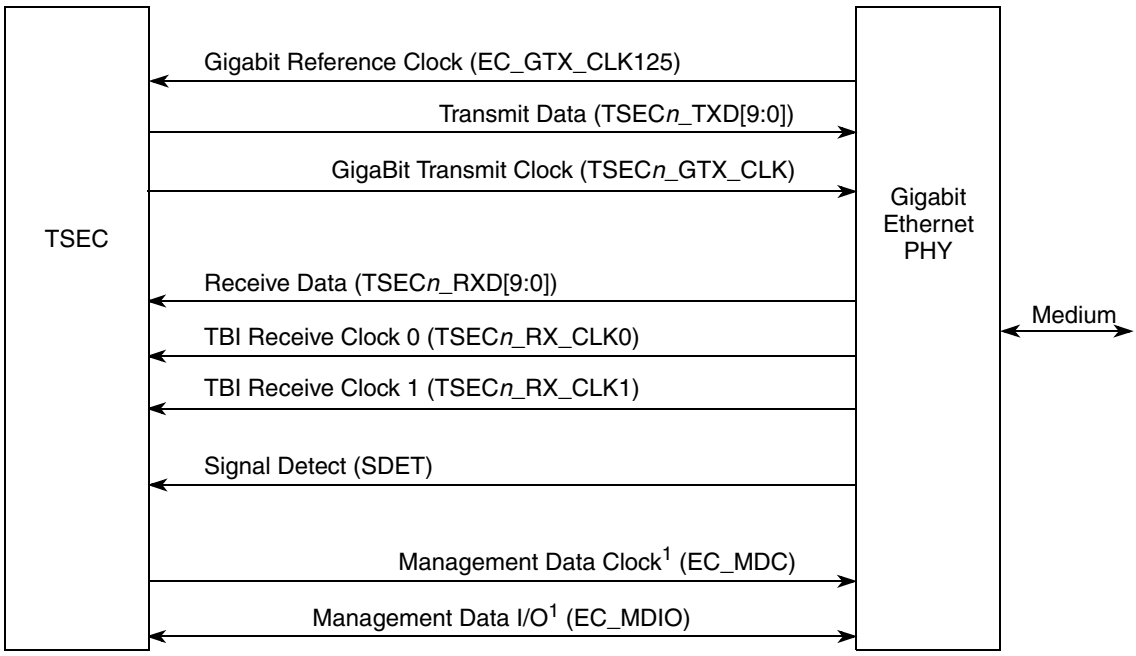


[1] The management signals (EC_MDC and EC_MDIO) are common to all of the gigabit Ethernet controllers' module connections in the system, assuming that each PHY has a different management address.

Figure 14-112. TSEC-RGMII Connection

14.6.1.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) intended to be used between the PHYs and the TSEC to implement a standard SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 14-113 shows the basic components of the TBI including the signals required to establish TSEC module connection with a PHY. RBC0 and RBC1 are differential 62.5-MHz receive clocks.



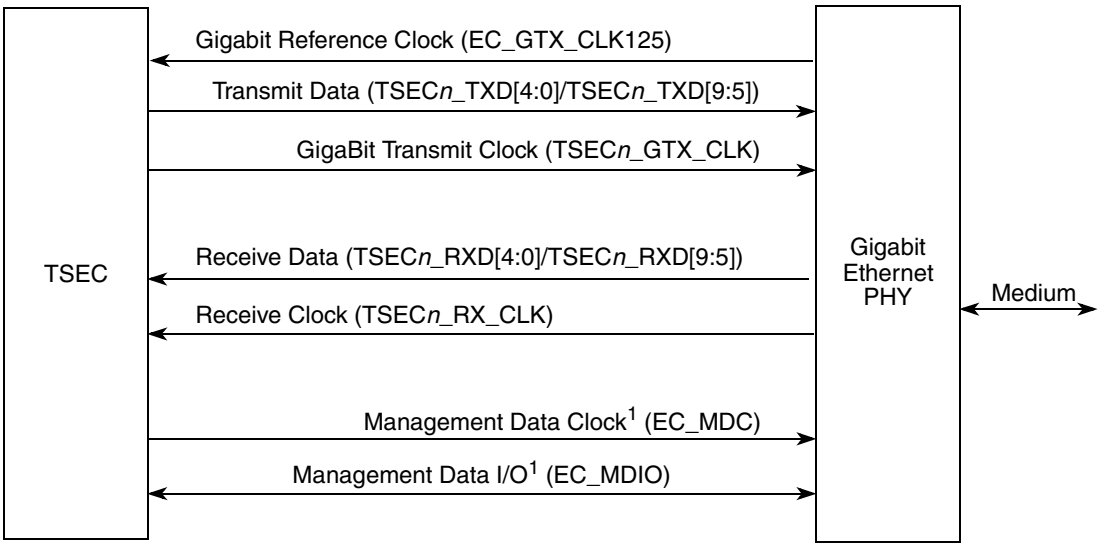
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-113. TSEC-TBI Connection

A TBI interface has 27 signals (EC_GTX_CLK125 included) for connecting to an Ethernet PHY, as defined by IEEE 802.3z GMII and TBI standards.

14.6.1.5 Reduced Ten-Bit Interface (RTBI)

This section describes the reduced ten-bit interface (RTBI) intended to be used between the PHYs and the TSEC to implement a reduced signal count version of a SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 14-114 shows the basic components of the RTBI including the signals required to establish TSEC module connection with a PHY.



[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-114. TSEC-RTBI Connection

A RTBI interface has 15 signals (EC_GTX_CLK125 included), as defined by the RGMII specification Version 1.2a 9/22/00, and is intended to be an alternative to the IEEE 802.3u MII, the IEEE 802.3z GMII and the TBI standard for connecting to an Ethernet PHY.

Table 14-110 describes the signal multiplexing for GMII, MII and TBI interfaces.

Table 14-110. GMII, MII, and TBI Signal Multiplexing

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII Interface Frequency [MHz] 125 Voltage[V] 3.3			MII Interface Frequency [MHz] 25 Voltage[V] 3.3			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1				GTX_CLK	O	1
TX_CLK	I	1	TX_CLK	I	1	TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TxD[2]	O	1	TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TxD[3]	O	1	TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TxD[4]	O	1				TCG[4]	O	1
TxD[5]	O	1	TxD[5]	O	1				TCG[5]	O	1
TxD[6]	O	1	TxD[6]	O	1				TCG[6]	O	1
TxD[7]	O	1	TxD[7]	O	1				TCG[7]	O	1
TX_EN	O	1	TX_EN	O	1	TX_EN	O	1	TCG[8]	O	1

Table 14-110. GMII, MII, and TBI Signal Multiplexing (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII Interface Frequency [MHz] 125 Voltage[V] 3.3			MII Interface Frequency [MHz] 25 Voltage[V] 3.3			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
TX_ER	O	1	TX_ER	O	1	TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RxD[2]	I	1	RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RxD[3]	I	1	RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RxD[4]	I	1				RCG[4]	I	1
RxD[5]	I	1	RxD[5]	I	1				RCG[5]	I	1
RxD[6]	I	1	RxD[6]	I	1				RCG[6]	I	1
RxD[7]	I	1	RxD[7]	I	1				RCG[7]	I	1
RX_DV	I	1	RX_DV	I	1	RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RX_ER	I	1	RX_ER	I	1	RCG[9]	I	1
COL	I	1				COL	I	1			
CRS	I	1				CRS	I	1	SDET	I	1
Sum		25	**Sum**		23	**Sum**		16	**Sum**		24

Table 14-111 describes the signal multiplexing for RGMII and RTBI interfaces.

Table 14-111. RGMII and RTBI Signal Multiplexing

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII Interface Frequency [MHz] 125 Voltage[V] 2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1						
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]/TCG[8]	O	1

Table 14-111. RGMII and RTBI Signal Multiplexing (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII Interface Frequency [MHz] 125 Voltage[V] 2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
TxD[4]	O	1						
TxD[5]	O	1						
TxD[6]	O	1						
TxD[7]	O	1						
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1						
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1						
RxD[5]	I	1						
RxD[6]	I	1						
RxD[7]	I	1						
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1						
COL	I	1						
CRS	I	1						
Sum		25	**Sum**		12	**Sum**		12

Table 14-112 describes the signals shared by all interfaces.

Table 14-112. Shared Signals

Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	Reference clock
Sum		3	

14.6.2 Gigabit Ethernet Channel Operation

This section describes the operation of the TSEC. First, the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is reviewed. Address recognition and hash table algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loopback descriptions.

14.6.2.1 Initialization Sequence

This section describes which registers are reset due to a hard or software reset and what registers the user must initialize prior to enabling the TSEC.

14.6.2.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All TSEC registers and control logic are reset to their default states after a hard reset has occurred.

14.6.2.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic TSEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. The module memory map in Table 14-3 lists all the TSEC registers. Table 14-113 describes the minimum steps for register initialization.

Table 14-113. Steps of Minimum Register Initialization

Description
1. Set, then clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII Mgmt interface
5. Configure the TBI control to TBI or GMII
6. Clear IEVENT

Table 14-113. Steps of Minimum Register Initialization (continued)

Description
7. Initialize IMASK
8. Initialize IADDR*n*
9. Initialize GADDR*n*
10. Initialize RCTRL
11. Initialize DMACTRL

After the registers are initialized, the user must execute the following steps in the order described below to bring the TSEC into a functional state (out of reset):

1. For the transmission of Ethernet frames, TxBDs must first be built in memory, linked together as a ring, and pointed to by the TBASE register. A minimum of two buffer descriptors per ring is required. Setting the ring to a size of one causes the same frame to be transmitted twice.
2. Likewise, for the reception of Ethernet frames, the receive queue must be ready, with its RxBD pointed to by the RBASE register. Both transmit and receive can be gracefully stopped after transmission and reception begins.
3. Write to MACCFG1 and set the appropriate bits. These need to include Rx_EN and Tx_EN. To enable flow control, Rx_Flow and Tx_Flow must also be set.
4. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. DMACTRL[GRS] must be cleared if the receiver had been previously stopped. See 14.5.3.1.7, “DMA Control Register (DMACTRL),” and Section 14.6.3.1, “Transmit Data Buffer Descriptor (TxBD),” for more information.

14.6.2.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft reset to and/or reconfiguring the MAC with new parameters, user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. User must gracefully stop the DMA by setting both GRS and GTS bits in the DMACTRL register, then wait for both GRSC and GTSC bits to be set in the IEVENT register before resetting the MAC or changing parameters. Both GRS and GTS bits must be cleared before re-enabling the MAC to resume the DMA.

During the MAC configuration, if a new set of Tx buffer descriptors are used, the user must load the pointers into the TBASE register. Likewise if a new set of Rx buffer descriptors are used, the RBASE register must be written with the new pointer.

Following is a procedure to gracefully reset and reconfigure the MAC:

1. Set GTS bit in DMACTRL register
2. Poll GTSC bit in IEVENT register until detected as set
3. Clear both Rx_EN and Tx_EN bits in MACCFG1
4. Wait for a period of 9.6 Kbytes worth of data on the interface (~8 ms worst case).
5. Set GRS bit in DMACTRL register
6. Poll GRSC bit in IEVENT register until detected as set
7. Set Soft_Reset bit in MACCFG1 register

8. Clear Soft_Reset bit in MACCFG1 register
9. Load TBASE with new TxBD pointer
10. Load RBASE with new RxBD pointer
11. Set up other MAC registers (MACCFG2, MAXFRM, etc.)
12. Set WWR and WOP bits in DMACTRL register
13. Clear THLT bit in TSTAT register and QHLT bit in RSTAT register by writing 1 to these bits.
14. Clear GRS/GTS bits in DMACTRL (do not change other bits)
15. Enable Tx_EN/Rx_EN in MACCFG1 register

14.6.2.3 Gigabit Ethernet Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the TSEC begins to poll the first transmit buffer descriptor (TxBD) in the TxBD ring every 512 transmit clocks. If TxBD[R] is set, TSEC begins moving transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the GMII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, a transmit-on-demand function may be emulated while in polling mode by using the graceful-transmit-stop feature. First, clear the IMASK[GTSCEN] bit to mask the graceful-transmit-stop complete interrupt. Next set, then immediately clear the DMACTRL[GTS] bit. Clear the resulting IEVENT[GTSC] bit. Finally, the IMASK[GTSCEN] bit may be set once again.There is one internal buffer for out-of-sequence flow control frames. While the TSEC is between frames, this buffer is polled. The buffer must contain the whole frame. Once the TSEC is in paused mode, the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC regards it as a regular TxBD.

In half-duplex mode (MACCFG2[Full Duplex] is cleared) the MAC defers transmission if the line is busy (CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if CRS remains negated for 60 clocks. If so, transmission begins after an additional 36 bit times (96 bit times after CRS originally became negated). If CRS continues to be asserted, the MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in the Tx FIFO is re-transmitted in case of a collision. This improves bus usage and latency.

The transmitter also monitors for an abort condition and terminates the current frame if an abort condition is encountered. In full-duplex mode the protocol is independent of network activity, and only the transmit inter-frame gap must be enforced.

The transmit block also implements full-duplex flow control. If a flow control frame is received, the MAC does not service the transmitter’s request to send data until the pause duration is over. If the MAC is currently sending data when a pause frame is received, the MAC finishes sending the current frame, then suspends subsequent frames (except a pause frame) until the pause duration is over. The pause duration is defined by the received pause control frame and begins when the frame was first received. In addition, the transmitter supports transmission of flow control frames through TCTRL[TFC_PAUSE]. The transmit

pause frame is generated internally based on the PAUSE register that defines the pause value to be sent. Note that it is possible to send a pause frame while the pause timer has not expired.

The MAC automatically appends FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] is set in first TxBD
- TxBD[TC] is set in first TxBD
- MACCFG2[PAD/CRC] is set
- MACCFG2[CRC EN] is set

Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the current buffer is reached and TxBD[L] is cleared (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD/CRC] is set, the Ethernet controller pads any frame shorter than 64 bytes.

To pause transmission or rearrange the transmit queue, set DMACTRL[GTS]. This can help in transmitting expedited data ahead of previously linked buffers or for error situations. If GTS is set, the TSEC transmitter performs a graceful transmit stop. The Ethernet controller stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with an error. The IEVENT[GTSC] interrupt occurs once the graceful transmit stop operation is completed. After GTS is cleared, the TSEC resumes transmission with the next frame.

While the TSEC is in 10/100 Mbps mode it sends bytes least-significant nibble first and each nibble is sent lsb first. While it is in 1-Gbps mode it sends bytes lsb first.

14.6.2.4 Gigabit Ethernet Frame Reception

The TSEC Ethernet receiver is designed to work with little core intervention and can perform address recognition, CRC checking, short frame checking, and maximum frame-length checking. The receiver can also force frame headers and buffer descriptors to be allocated into the L2 cache. See Section 14.6.4, “Data Extraction to the L2 Cache,” for additional information.

After a hardware reset, the software driver clears the RSTAT register and sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames. If TSEC*n*_RX_DV is asserted and TSEC*n*_COL remains negated, the MAC strips a valid preamble/SFD (start of frame delimiter) header and begins processing the frame. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the TSEC controller begins to perform the frame recognition function through destination address (DA) recognition (See Section 14.6.2.6, “Frame Recognition,” for additional information.). Based on this match the frame can be accepted or rejected. Once accepted, the TSEC processes the frame based on user-defined attributes.

The receiver can also filter frames based on physical (individual), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame recognition algorithm is complete, system bus usage is not wasted on frames unwanted by this station.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from the queue. If RxBD is not being used by software (RxBD[E] is set), the TSEC starts transferring the incoming frame. RxBD[F] is set for the first RxBD used for any particular receive frame.

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

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBL). The smallest valid value is 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. After the frame ends (CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

Receive frames are not truncated if they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame] is set, yet the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

After the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates an interrupt (that can be masked) indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception or rearrange the receive queue, DMACTRL[GRS] must be set. If this bit is set, the TSEC receiver performs a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event is signalled after the graceful receive stop operation is completed. While in this mode the user can then clear IEVENT[GRSC] and can write to registers that are accessible to both the user and the TSEC hardware without fear of conflict. After DMACTRL[GRS] is cleared, the TSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter), it resumes receiving, and the first valid frame received is placed in the next available RxBD.

14.6.2.5 RMON Support

TSEC automatically gathers network statistics required for RMON without needing to receive all addresses. The RMON MIB group 1, RMON MIB group 2, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB are supported.

For RMON statistics and their corresponding counters see the memory map.

14.6.2.6 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

14.6.2.6.1 Destination Address Recognition

The Ethernet controller can also perform the frame filtering using the traditional destination address (DA) recognition methods.

Figure 14-115 is a flowchart for address recognition on received frames that is used to explain the concept. In the actual implementation most of the decision points shown in the figure actually occur simultaneously.

The Ethernet controller compares the destination address field of the received frame with the physical address the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, then the controller performs address recognition on multiple individual addresses using the IADDR*n* hash table. The user must write zeros to the hash in order to avoid a hash match, and ones to station address in order to avoid individual address match, or the user can turn on promiscuous mode. (See section Section 14.5.3.4.1, “Receive Control Register (RCTRL).”)

In the group type of address recognition, the Ethernet controller determines whether the group address is a broadcast address. If it is a broadcast, and broadcast addresses are enabled, the frame is accepted. If the group address is not a broadcast address, the user can perform address recognition on multiple group addresses using the GADDR*n* hash table. In promiscuous mode, the Ethernet controller receives all of the incoming frames regardless of their address.

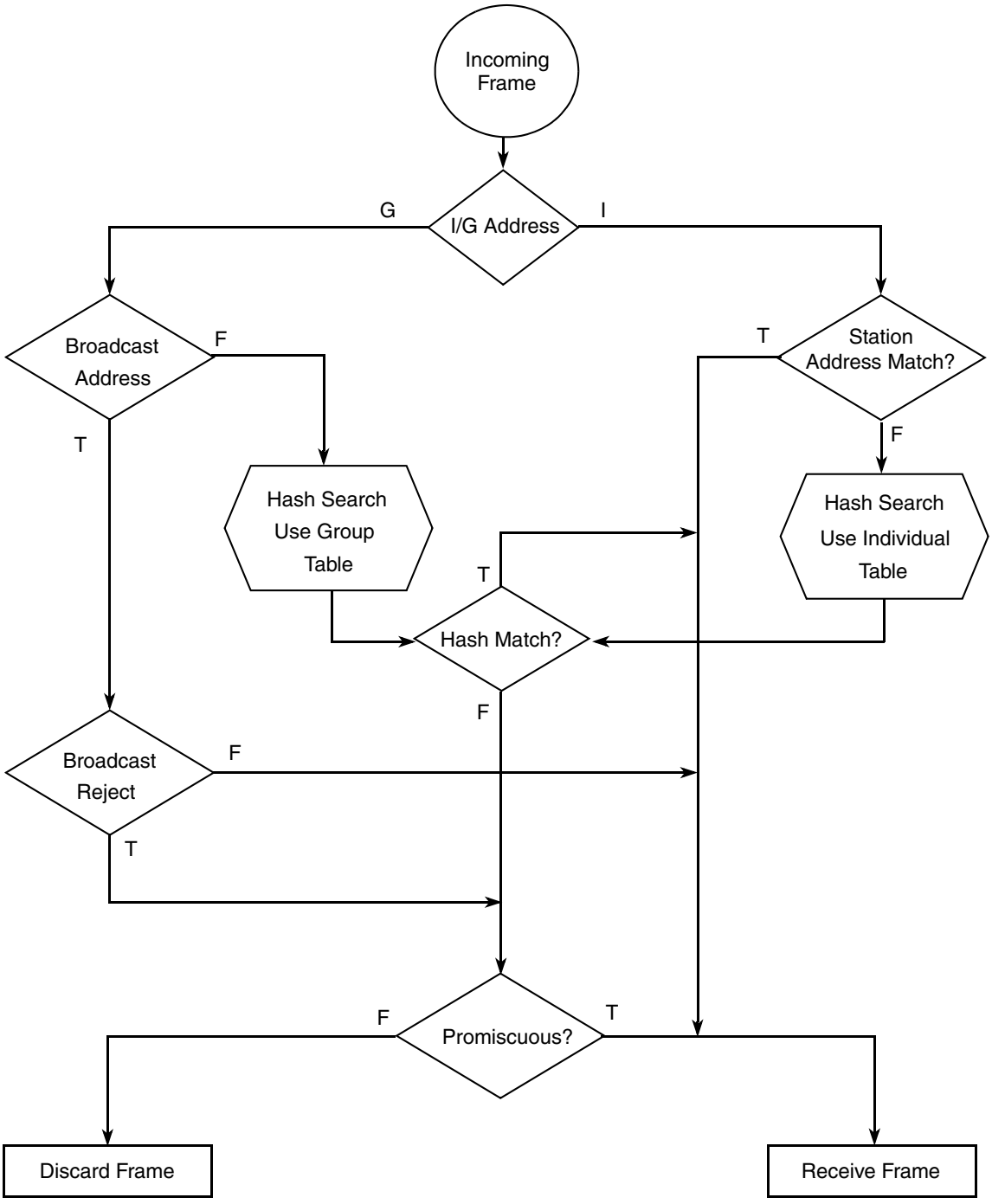


Figure 14-115. Ethernet Address Recognition Flowchart

14.6.2.6.2 Hash Table Algorithm

The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in GADDR0–7 or IADDR0–7. The eight high-order bits of a cyclix redundancy check (CRC) checksum are used to index into the hash table. The high-order three bits of this 8-bit field are used to select one of the eight registers in either the individual or group hash table. The low-order five bits select a bit within the 32-bit register. A value of 0 in the high-order three bits selects IADDR0/GADDR0.

The same process is used if the Ethernet controller receives a frame. If the CRC checksum selects a bit that is set in the group/individual hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 224/256 (87.5%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if 32 group and 32 physical addresses are stored in their respective hash tables, because 87.5% of all group addresses and 87.5% of all individual address are rejected, then 87.5% of all frames are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 256-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

14.6.2.6.3 CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value for the purposes of TSEC. The RFC 3309 algorithm uses the following polynomial to calculate the CRC value:

x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x1 + x0 or 0x04c11db7

Given a destination MAC address of DA = 01000CCCCCCC, the algorithm results in a CRC remainder value of 0xA29F4BBC.

Bit-reversing the low-order byte of the CRC value (0xBC) yields:

BR_CRC = 0x3D = 0b00111101

The high-order 3-bits of the new BR_CRC value are used to select which 32-bit register (of the 8) to use. This example maps the DA to register 1.

High-order 3 bits of BR_CRC: HO_CRC = 0b001 = 1

The low-order 5 bits are used to select which bit to set in the given register (with a value of 0 setting 0x8000_0000 and 31 setting 0x0000_0001). Therefore, the example DA maps to bit 29 of register 1.

Low-order 5 bits of BR_CRC: LO_CRC = 0b11101 = 29

Therefore, GADDR1 is ORed with the value 0x0000_0004.

Additional calculated examples follow:

Example 1:

- Destination MAC address: DA = 01005E000128
- CRC remainder value: CRC = 0x821D6CD3
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xCB = 0b11001011
- High-order 3 bits of BR_CRC: HO_CRC = 0b110 = 6
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01011 = 11
- GADDR6 = 0x0010_0000

Example 2:

- Destination MAC address: DA = 0004F0604F10
- CRC remainder value: CRC = 0x1F5A66B5
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xAD = 0b10101101
- High-order 3 bits of BR_CRC: HO_CRC = 0b101 = 5
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01101 = 13
- GADDR5 = 0x0004_0000

14.6.2.7 Flow Control

Because collisions cannot occur in full-duplex mode, gigabit Ethernet can operate at the maximum rate. If the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 14-114 lists the flow-control frame structure.

Table 14-114. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble		
1	SFD		Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames
6	Source address		
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter		Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed-independent constant of 512 bit-times (unlike slot time). The most-significant octet is sent first.
2	Extended MAC parameter		Extended pause control parameter as defined by the PTV[PTE] field. The most-significant octet is sent first.
40	Reserved	—	
4	FCS		Frame check sequence (CRC)

If flow-control mode is enabled (MACCFG1[Rx_Flow] is set) and the receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. During this pause, only a control frame can be sent (TCTRL[TFC_PAUSE] is set). Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

14.6.2.8 Interrupt Handling

The following describes what usually occurs within a TSEC interrupt handler:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time.
- Process the TxBDs to reuse them if the IEVENT[TXB or TXF] were set. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the TSEC; thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set. See Table 14-115.
- Obtain data from the RxBD if IEVENT[RXC,RXB or RXF] is set. If the receive speed is fast or the interrupt delay is long, the TSEC may have received more than one RxBD; thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the TSEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch. See Table 14-116.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 14-115. Non-Error Transmit Interrupts

Interrupt	Description	Action Taken by TSEC
GTSC	Graceful transmit stop complete: transmitter is put into a pause state after completion of the frame currently being transmitted.	None
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.	None
TXB	Transmit buffer: A transmit buffer descriptor, that is not the last one in the frame, was updated.	Programmable ‘write with response’ TxBD to memory before setting IEVENT[TXB].
TXF	Transmit frame: A frame was transmitted and the last transmit buffer descriptor (TxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last TxBD before setting IEVENT[TXF].

Table 14-116. Non-Error Receive Interrupts

Interrupt	Description	Action Taken by TSEC
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame currently being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was first received.	None
RXB	Receive buffer: A receive buffer descriptor, that is not the last one of the frame, was updated.	Programmable 'write with response' RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame was received and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable 'write with response' to memory on the last RxBD before setting IEVENT[RXF].

14.6.2.8.1 Interrupt Coalescing

Interrupt coalescing offers the user the ability to contour the behavior of the TSEC with regard to frame interrupts. Separate but identical mechanisms exist for handling both transmitted frames and received frames. While interrupt coalescing is enabled a transmit or receive frame interrupt (resulting from the interrupt bit (I) of the buffer descriptor in question and appropriately-enabled by the IMASK register) is raised either when a counter threshold-defined number of frames is received/transmitted or the timer threshold-defined period of time has lapsed, whichever occurs first.

14.6.2.8.2 Interrupt Coalescing By Frame Count Threshold

To avoid interrupt bandwidth congestion due to frequent, consecutive interrupts, the user may enable and configure interrupt coalescing to deliberately group frame interrupts, reducing the total number of interrupts raised. The number of frames received or transmitted prior to an interrupt being raised is determined by the frame threshold field (ICFCT) in the appropriate interrupt coalescing configuration register (RXIC or TXIC). The frame threshold field may be assigned a value between 1 and 255. A value of 0 results in boundedly undefined behavior. Note that a value of 1 functionally defeats the advantages of interrupt coalescing since the frame threshold is reached with each frame received or transmitted. Once the number of frames transmitted or received reaches the threshold limit, an interrupt is raised (if the interrupt bit of the frame buffer descriptor(s) is set and if appropriately-enabled in the IMASK register), the threshold counter is reset, and then continues counting frames while the interrupt is active. The threshold counter is also reset if an interrupt is raised subject to the corresponding threshold timer.

14.6.2.8.3 Interrupt Coalescing By Timer Threshold

To avoid stale frame interrupts, the user may also assign a timer threshold, beyond which any frame interrupts not yet raised are forced (if the interrupt bit of the frame buffer descriptor(s) is set and if enabled in the IMASK register). The timer threshold fields of the receive and transmit interrupt coalescing configuration registers (RXIC[ICTT] and TXIC[ICTT]) are defined in units equivalent to 64 TSEC interface clocks. That is, one timer threshold unit is 26.5 µs, 2.56 µs, or 512 ns, corresponding to interface modes 10 Mbps, 100 Mbps, or 1 Gbps, respectively.

After transmitting a frame (with TXBD[I] set, causing IEVENT[TXF] to be set), the transmit interrupt coalescing threshold timer begins counting. When the timer threshold has been reached an interrupt is raised if the interrupt bit of the buffer descriptor is set and IMASK[TXFEN] is set.

After receiving a frame (with RXBD[I] set, causing IEVENT[RXF] to be set), the receive interrupt coalescing threshold timer begins counting. When the timer threshold has been reached an interrupt is raised if the interrupt bit of the buffer descriptor is set and IMASK[RXFEN] is set.

The interrupt coalescing timer thresholds (transmit and receive, operating independently) may be values ranging from 1 to 65535. A value of 0 is illegal and results in behavior identical to that when interrupt coalescing is disabled (corresponding RXIC[ICEN] or TXIC[ICEN] is cleared). Table 14-117 specifies the range of possible timing thresholds subject to the interface frequency and the value of RXIC[ICTT] or TXIC[ICTT].

Table 14-117. Interrupt Coalescing Timing Threshold Ranges

TSEC Interface Format and Frequency	Interrupt Coalescing Threshold Time	
	Minimum (ICTT = 1)	Maximum (ICTT = 65535)
10Base-T at 2.5 MHz	25.6 µs	1.678 s
100Base-T at 25 MHz	2.56 µs	167.8 ms
1000Base-T at 125 MHz	512 ns	33.55 ms

The transmit or receive timer threshold counter is reset when a corresponding interrupt is raised and begins counting again upon deassertion of that interrupt and once IEVENT[TXF or RXF] is set.

14.6.2.9 Inter-Packet Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-packet gap). After a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN and then begins retransmission on the LAN. This process is called a retry. If the packet is not successfully sent within a specified number of retries, an error is indicated.

The minimum inter-packet gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back packets with this minimum spacing. In addition, after waiting a required number of clocks (based on the back-off algorithm), the transmitter waits for carrier sense to be negated before retransmitting the packet. Retransmission begins 36 serial clocks after carrier sense is negated for at least 60 serial clocks.

14.6.2.10 Internal and External Loopback

Setting MACCFG1[Loop Back] causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. This bit is cleared by default.

14.6.2.11 Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Programming note: When the TSEC encounters a halt condition (TSTAT[THLT] is set), it stops processing the frame at the current TxBD. The TSEC relies on the user to manage the buffer descriptor pointer, TBPTR, or the buffer descriptor queue before resuming transmissions. Once the TSEC resumes, it fetches the TxBD pointed to by TBPTR.

Transmission errors are described in Table 14-118.

Table 14-118. Transmission Errors

Error	Response
Transmitter underrun	The controller sends 32 bits that ensure a CRC error, terminates buffer transmission, sets TxBD[UN], closes the buffer, sets IEVENT[XFUN] and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, sets IEVENT[CRL/XDA] and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Excessive defer abort	The controller terminates buffer transmission, sets TxBD[DEF], closes the buffer, sets IEVENT[CRC/XDA], and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared.
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, sets IEVENT[LC] and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Memory Read Error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], DMA stops sending data to the FIFO which causes an underrun error but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits are continued once TSTAT[THLT] is cleared.
Babbling Transmit Error	A frame is transmitted which exceeds the MAC's maximum frame length and MACCFG2[Huge Frame] is a 0. The controller sets IEVENT[BABT] and continues without interruption. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame.

Reception errors are described in Table 14-119.

Table 14-119. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxBD[OV], sets RxBD[L], closes the buffer, and sets IEVENT[RXF], The receiver then enters hunt mode (seeking start of a new frame).
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY]. In addition, the RSTAT[QHLT] bit is set. The halted queue resumes reception once the user clears the RSTAT[QHLT] bit.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The TSEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.

Table 14-119. Reception Errors (continued)

Error	Description
CRC error	If a CRC error occurs, the controller sets RxBD[CR], closes the buffer, and sets IEVENT[RXF]. This TSEC relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.
Memory Read Error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR] and discards the frame. In addition the RSTAT[QHLT] bit is set. The halted queue resumes reception once the RSTAT[QHLT] bit is cleared.
Babbling Receive Error	A frame is received that exceeds the MAC's maximum frame length. The controller sets IEVENT[BABR] and continues

14.6.3 Buffer Descriptors

The TSEC buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller BD for ease of reuse. Drawing from the MPC8260 FEC BD programming model, the TSEC descriptor base registers point to the beginning of BD rings. The 8-byte data BD format is similar to the MPC8260 BD model.

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 14-116). Data BDs encapsulate all information necessary for the TSEC to transmit or receive an Ethernet frame. Within each data BD there is a status field, a data length field, and a data pointer. The BD completely describes an Ethernet packet by centralizing status information for the data packet in the status field of the BD and by containing a data BD pointer to the location of the data buffer. Software is responsible for setting up the BDs in memory. Because of pre-fetching, a minimum of two buffer descriptors per ring are required. This applies to both the transmit and the receive descriptor rings. Software also must have the data pointer pointing to memory. Within the status field, there exists an ownership bit which defines the current state of the buffer (pointed to by the data pointer). Other bits in the status field of the buffer descriptor are used to communicate status/control information between the TSEC and the software driver.

The status field of the BD is 16-bit field, as is the length field. The data buffer pointer is a 32-bit field. Therefore, the BDs should be accessed with the following C structure:

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct bd_struct {
   uint_16 flags;
   uint_16 length;
   uint_32 bufptr;
};
```

Because there is no next BD pointer in the transmit/receive BD (see Figure 14-117), all BDs must reside sequentially in memory. The TSEC increments the current BD location appropriately to the next BD location to be processed. There is a wrap bit in the last BD that informs the TSEC to loopback to the beginning of the BD chain. Software must initialize TBASE and RBASE that point to the beginning transmit and receive BDs for TSEC.

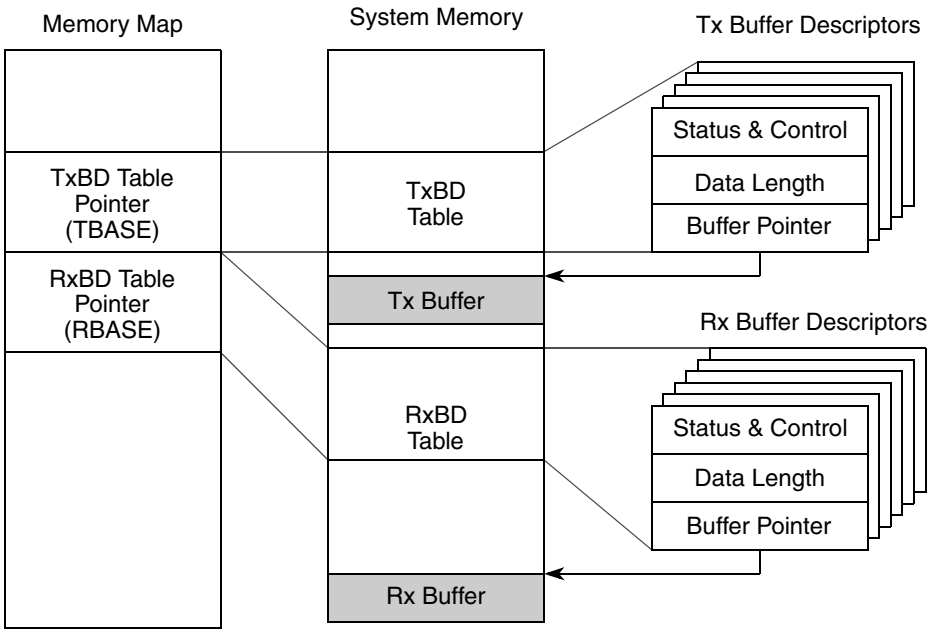


Figure 14-116. Example of TSEC Memory Structure for BD

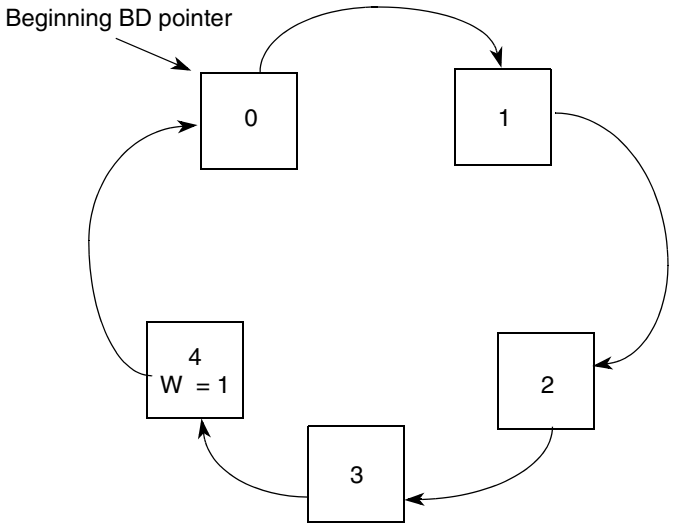


Figure 14-117. Buffer Descriptor Ring

14.6.3.1 Transmit Data Buffer Descriptor (TxBD)

Data is presented to the TSEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD/CRC, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The TSEC clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block. Figure 14-118 shows the TxBD.

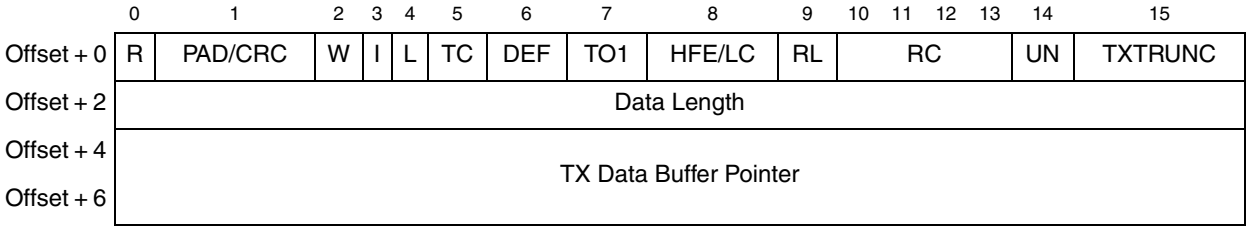


Figure 14-118. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 14-120.

Table 14-120. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	R	Ready. Written by TSEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The TSEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which is prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
Offset + 0	1	PAD/CRC	PAD/CRC. Padding and CRC attachment for frames. (Valid only while it is set in the first BD and MACCFG2[PAD/CRC] is cleared.) If MACCFG2[PAD/CRC] is set, this bit is ignored. 0 Do not add padding to short frames. No CRC is appended unless TxBD[TC] is set. 1 Add PAD/CRCs to frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADs up to MINFLR value, TSEC PADs always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames.
Offset + 0	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in TBASE.
Offset + 0	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
Offset + 0	4	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
Offset + 0	5	TC	Tx CRC. Written by user. (Valid only while it is set in first BD and TxBD[PAD/CRC] is cleared and MACCFG2[PAD/CRC] is cleared and MACCFG2[CRC EN] is cleared.) If MACCFG2[PAD/CRC] is set or MACCFG2[CRC EN] is set, this bit is ignored. 0 End transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.

Table 14-120. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	6	DEF	Defer indication. Hardware updates this bit if an excessive defer condition occurs. 0 This frame was not deferred. 1 If HAFDUP[EXCESS_DEFER] = 1, this frame did not have a collision before it was sent but it was sent late because of deferring. If HAFDUP[EXCESS_DEFER] = 0, this frame was aborted and not sent.
Offset + 0	7	TO1	Transmit software ownership. This read/write bit may be utilized by software, as necessary. Its state does not affect the hardware nor is it affected by the hardware.
Offset + 0	8	HFE/LC	Huge frame enable (written by user)/Late collision (written by TSEC) Huge frame enable. Written by user. Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's Maximum Frame Length register. 1 Do not truncate the transmit frame. Late collision. Written by TSEC. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The TSEC terminates the transmission and updates LC.
Offset + 0	9	RL	Retransmission Limit. Written by TSEC. 0 Transmission before maximum retry limit is hit. 1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The TSEC terminates the transmission and updates RL.
Offset + 0	10–13	RC	Retry Count. Written by TSEC. 0 The frame is sent correctly the first time or if RL is set then the retry limit has been reached x One or more attempts were needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
Offset + 0	14	UN	Underrun. Written by TSEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The TSEC terminates the transmission and updates UN.
Offset + 0	15	TXTRUNC	TX truncation. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for the frame.
Offset + 2	0–15	Data Length	Data length is the number of octets the TSEC transmits from this BD's data buffer. It is never modified by the TSEC. This field must be greater than zero.
Offset + 4	0–31	TX Data Buffer Pointer	The transmit buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

14.6.3.2 Receive Buffer Descriptor (RxBD)

In the RxBD the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the TSEC modifies the E, L, F, M, BC, MC, LG, NO, SH, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, SH, CR, OV,

and TR bits in the first word of the buffer descriptor are only modified by the TSEC if the L bit is set. The first word of the RxBD contains control and status bits. Its format is detailed in Figure 14-119.

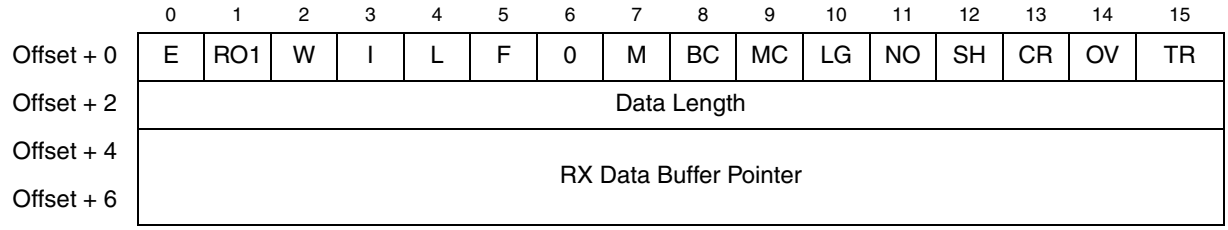


Figure 14-119. Receive Buffer Descriptor

Table 14-121 describes the fields of the RxBD.

Table 14-121. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty. Written by TSEC (when cleared) and by user (when set). 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1	RO1	Receive software ownership bit. Reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.
Offset + 0	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] is set or IMASK[RXFEN] is set). If the user wants to be interrupted only if RXF occurs, then the user must disable RXB (IMASK[RXBEN] is cleared) and enable RXF (IMASK[RXFEN] is set).
Offset + 0	4	L	Last in frame. Written by TSEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	5	F	First in frame. Written by TSEC. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
Offset + 0	6	—	Reserved
Offset + 0	7	M	Miss. Written by TSEC. (This bit is valid only if the L-bit is set and TSEC is in promiscuous mode.) This bit is set by the TSEC for frames that were accepted in promiscuous mode, but were flagged as a ‘miss’ by the internal address recognition; thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	8	BC	Broadcast. Written by TSEC. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
Offset + 0	9	MC	Multicast. Written by TSEC. (Only valid if L is set.) Is set if the DA is multicast and not BC.

Table 14-121. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	10	LG	Rx frame length violation. Written by TSEC. (Only valid if L is set.) A frame length greater than maximum frame length was recognized while MACCFG2[Huge Frame] was set. Note, if MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the Maximum Frame Length register.
Offset + 0	11	NO	Rx non-octet aligned frame. Written by TSEC. (Only valid if L is set.) A frame that contained a number of bits not divisible by eight was received.
Offset + 0	12	SH	Short frame. Written by TSEC. (only valid if L is set.) A frame length that was less than the minimum length defined for this channel (MINFLR) was recognized, provided RCTRL[RSF] is set.
Offset + 0	13	CR	Rx CRC error. Written by TSEC. (Only valid if L is set.) This frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected.
Offset + 0	14	OV	Overrun. Written by TSEC. (Only valid if L is set.) A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, and CR lose their normal meaning and are zero.
Offset + 0	15	TR	Truncation. Written by TSEC. (Only valid if L is set.) Is set if the receive frame is truncated. This can happen if a frame length greater than maximum frame length was received and the MACCFG2[Huge Frame] is cleared. If this bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	0–15	Data Length	Data length. Written by TSEC. Data length is the number of octets written by the TSEC into this BD's data buffer if L is cleared (the value is equal to MRBL), or the length of the frame including CRC, if L is set.
Offset + 4	0–31	Rx Data Buffer Pointer	Receive buffer pointer. Written by user. The receive buffer pointer, which always points to the first location of the associated data buffer, must be 64-byte aligned. The buffer must reside in memory external to the TSEC.

14.6.4 Data Extraction to the L2 Cache

Some applications require the ability to identify selected portions of data within a frame's data payload. This process is called extraction; although, the data is not truly removed. Rather than literally extracting a section of the data and copying it into a new memory location, the data is placed in the L2 cache. This allows the processor to quickly access critical frame information as soon as the processor is ready without having to first fetch the data from main memory. This results in substantial improvement in throughput and hence reduction in latency.

Extraction functionality is controlled and configured with ATTR and ATTRELI. See Section 14.5.3.9.1, "Attribute Register (ATTR)," and Section 14.5.3.9.2, "Attribute Extract Length and Extract Index Register (ATTRELI)," for specific register information.

14.7 Initialization/Application Information

14.7.1 Interface Mode Configuration

This section describes how to configure the TSEC in different supported interface modes. These include MII, GMII, TBI, RGMII, RTBI. The pinout, the data registers that must be initialized, as well as speed selection options are described. The ECNTRL[TBIM] and ECNTRL[RPM] bits are written, assuming the part was not pin-configured at initialization to the correct mode.

14.7.1.1 MII Interface Mode

Table 14-122 describes the signal configurations required for MII interface mode.

Table 14-122. MII Interface Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			MII Interface Frequency [MHz] 25/2.5 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	Leave unconnected	O	
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	Leave unconnected	O	
TxD[5]	O	1	Leave unconnected	O	
TxD[6]	O	1	Leave unconnected	O	
TxD[7]	O	1	Leave unconnected	O	
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	Not used	I	
RxD[5]	I	1	Not used	I	
RxD[6]	I	1	Not used	I	
RxD[7]	I	1	Not used	I	
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1

Table 14-122. MII Interface Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			MII Interface Frequency [MHz] 25/2.5 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
COL	I	1	COL	I	1
CRS	I	1	CRS	I	1
Sum		25	**Sum**		16

Table 14-123 describes the shared signals of the MII interface.

Table 14-123. Shared MII Signals

TSEC Signals	I/O	No. of Signals	MII Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	Not used	I	0	Reference clock
Sum		3	**Sum**		2	

Table 14-124 describes the register initializations required to reconfigure the PHY to MII mode following initial auto-negotiation.

Table 14-124. MII Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled low for G/MII mode.
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half-duplex operation. Set I/F mode bit, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC station address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC station address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)

Table 14-124. MII Mode Register Initialization Steps (continued)

Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Reset the management interface. MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0000]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0101] set source clock divide by 14, for example, to insure that EC_MDC clock speed is approximately 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Setup the MII Mgmt for a write cycle to the external PHY auxiliary control and status register to configure the PHY through the management interface (overrides configuration signals of the PHY). MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY Writing to MII Mgmt control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Setup the MII Mgmt for a write cycle to the external PHY Extended PHY control register 1 to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Setup the MII Mgmt for a write cycle to the external PHY mode control register to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_00uu_00uu_0u00_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed auto-negotiation. Setup the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00.

Table 14-124. MII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of status register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MIIMSTAT register and check bit 10 (AN Done and Link is up) MIIMSTAT ---> [0000_0000_0000_0000_0000_000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK, (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n,* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n,* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL, (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL, (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) transmit descriptor ring and fill buffers with data. Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) receive descriptor ring and fill with empty buffers. Initialize RBASE, RBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

14.7.1.2 GMII Interface Mode

Table 14-125 describes the signal configurations required for GMII interface mode.

Table 14-125. GMII Interface Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII Interface Frequency [MHz] 125 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1

Table 14-125. GMII Interface Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII Interface Frequency [MHz] 125 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	TxD[4]	O	1
TxD[5]	O	1	TxD[5]	O	1
TxD[6]	O	1	TxD[6]	O	1
TxD[7]	O	1	TxD[7]	O	1
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	RxD[4]	I	1
RxD[5]	I	1	RxD[5]	I	1
RxD[6]	I	1	RxD[6]	I	1
RxD[7]	I	1	RxD[7]	I	1
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1			
CRS	I	1			
Sum		25	**Sum**		22

Table 14-126 describes the shared signals of the GMII interface.

Table 14-126. Shared GMII Signals

TSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	

Table 14-127 describes the register initializations required to reconfigure the PHY to GMII mode following initial auto-negotiation.

Table 14-127. GMII Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled low for G/MII mode.
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for GMII, full-duplex operation. Set I/F Mode bit. MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (This example has Full Duplex = 1, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC station address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC station address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a Physical address to the TBI so as to not conflict with the external PHY physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Reset the management interface, MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0000]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that EC_MDC clock speed is approximately 2.5 MHz.
Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Setup the MII Mgmt for a write cycle to the external PHY auxiliary control and status register to configure the PHY through the management interface (overrides configuration signals of the PHY), MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed
Setup the MII Mgmt for a write cycle to the external PHY extended PHY control register 1 to set up the interface mode selection MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]

Table 14-127. GMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Setup the MII Mgmt for a write cycle to the external PHY mode control register to set up the interface mode selection, MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_000u_00u1_0100_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed auto-negotiation. Setup the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00
Perform an MII Mgmt read cycle of status register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, Read the MIIMSTAT register and check bit 10 (AN Done and Link is up), MIIMSTAT ---> [0000_0000_0000_0000_0000_000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK, (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n,* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n,* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL, (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL, (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) transmit descriptor ring and fill buffers with data. Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]

Table 14-127. GMII Mode Register Initialization Steps (continued)

Initialize (Empty) receive descriptor ring and fill with empty buffers. Initialize RBASE, RBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

14.7.1.3 TBI Interface Mode

Table 14-128 describes the signal configurations required for TBI interface mode.

Table 14-128. TBI Interface Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TCG[4]	O	1
TxD[5]	O	1	TCG[5]	O	1
TxD[6]	O	1	TCG[6]	O	1
TxD[7]	O	1	TCG[7]	O	1
TX_EN	O	1	TCG[8]	O	1
TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RCG[4]	I	1
RxD[5]	I	1	RCG[5]	I	1
RxD[6]	I	1	RCG[6]	I	1
RxD[7]	I	1	RCG[7]	I	1
RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RCG[9]	I	1
COL	I	1		I	

Table 14-128. TBI Interface Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
CRS	I	1	SDET	I	1
Sum		25	**Sum**		24

Table 14-129 describes the shared signals for the TBI interface.

Table 14-129. Shared TBI Signals

TSEC Signals	I/O	No. of Signals	TBI Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	

Table 14-130 describes the register initializations required to reconfigure the PHY to TBI mode following initial auto-negotiation.

Table 14-130. TBI Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled high and EC_MDC signal pulled high for TBI mode. (TSEC attempts to auto-negotiate after system reset.)
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC station address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC station address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)

Table 14-130. TBI Mode Register Initialization Steps (continued)

Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0111] Set source clock divide by 28, for example, to ensure that EC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI control register (write the TBI address and register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.
Perform an MII Mgmt read cycle to verify state of TBI control register (optional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MIIMSTAT and look for AN Enable and other bit information.
Setup the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the link partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Setup the MII Mgmt for a write cycle to TBI's control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] the control register is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed auto-negotiation. Setup the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The Phy status control register is at address 0x1 and in this case the PHY address is 0x10.
Perform an MII Mgmt read cycle of status register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)

Table 14-130. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN expansion register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN link partner base page ability register. (optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 2 and 3 (Remote Fault) and bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n* (optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) transmit descriptor ring and fill buffers with data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) receive descriptor ring and fill with empty buffers Initialize RBASE, RBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

14.7.1.4 RGMII Interface Mode

Table 14-131 shows the signals configurations required for RGMII interface mode.

Table 14-131. RGMII Interface Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII Inrterface Frequency [MHz] 125 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1			
TxD[5]	O	1			
TxD[6]	O	1			
TxD[7]	O	1			
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1			
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1			
RxD[5]	I	1			
RxD[6]	I	1			
RxD[7]	I	1			
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1			
COL	I	1			
CRS	I	1			

Table 14-131. RGMII Interface Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII Inrterface Frequency [MHz] 125 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
Sum		25	**Sum**		12

Table 14-132 describes the shared signals for the RGMII interface.

Table 14-132. Shared RGMII Signals

TSEC Signals	I/O	No. of Signals	RGMII Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	

Table 14-133 describes the register initializations required to reconfigure the PHY to RGMII mode following initial auto-negotiation.

Table 14-133. RGMII Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled low and EC_MDC signal pulled low for RGMII mode. TBI control register's auto-negotiation enable and reset bits are ignored.
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has RGMII 10-Mbps mode, Statistics Enable = 1)
Initialize MAC station address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC station address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)

Table 14-133. RGMII Mode Register Initialization Steps (continued)

Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0111] Set source clock divide by 28, for example, to insure that EC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Setup the MII Mgmt for a write cycle to external the PHY AN advertisement register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Setup the MII Mgmt for a write cycle to the external PHY control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The Control register is at offset address 0x00 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt control with 16-bit data intended for the external PHY control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed auto-negotiation. Setup the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY status register is at address 0x1 and in this case the PHY Address is 0x2.
Perform an MII Mgmt read cycle of status register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN expansion register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]

Table 14-133. RGMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN link partner base page ability register. (optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) Transmit descriptor ring and fill buffers with data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) receive descriptor ring and fill with empty buffers Initialize RBASE, RBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

14.7.1.5 RTBI Interface Mode

Table 14-134 describes the signal configurations required for RTBI interface mode.

Table 14-134. RTBI Interlace Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TCG[1]/TCG[6]	O	1

Table 14-134. RTBI Interlace Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TxD[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1			
TxD[5]	O	1			
TxD[6]	O	1			
TxD[7]	O	1			
TX_EN	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1			
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1			
RxD[5]	I	1			
RxD[6]	I	1			
RxD[7]	I	1			
RX_DV	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1			
COL	I	1			
CRS	I	1		I	
Sum		25	**Sum**		12

Table 14-135 describes the shared signals for the RTBI interface.

Table 14-135. Shared RTBI Signals

TSEC Signals	I/O	No. of Signals	RTBI Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	

Table 14-136 describes the register initializations required to reconfigure the PHY to RTBI mode following initial auto-negotiation.

Table 14-136. RTBI Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled high and the EC_MDC signal pulled low for RTBI mode. (TSEC attempts to auto-negotiate after system reset.)
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC station address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0111] Set source clock divide by 28, for example, to insure that EC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Setup the MII Mgmt for a read cycle to TBI control register (write the TBI's address and register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.

Table 14-136. RTBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle to verify state of TBI control register (optional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MIIMSTAT and look for AN Enable and other bit information.
Setup the MII Mgmt for a write cycle to TBI's AN advertisement register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Write to MII Mgmt control with 16-bit data intended for TBI's AN advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Setup the MII Mgmt for a write cycle to TBI's control register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The control register is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed auto-negotiation. Setup the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The Phy status control register is at address 0x1 and in this case the PHY address is 0x10.
Perform an MII Mgmt read cycle of status register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN expansion register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MII Mgmt AN expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]

Table 14-136. RTBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN link partner base page ability register. (optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and full duplex) MII Mgmt AN link partner base page ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) transmit descriptor ring and fill buffers with data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) receive descriptor ring and fill with empty buffers Initialize RBASE, RBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

Chapter 15
DMA Controller

This chapter describes the DMA controller of the MPC8555E.

15.1 Introduction

The DMA controller transfers blocks of data between PCI, the local bus controller (LBC) interface, and the local address space, independent of the e500 core or external hosts.

15.1.1 Block Diagram

Figure 15-1 shows the block diagram of the DMA controller.

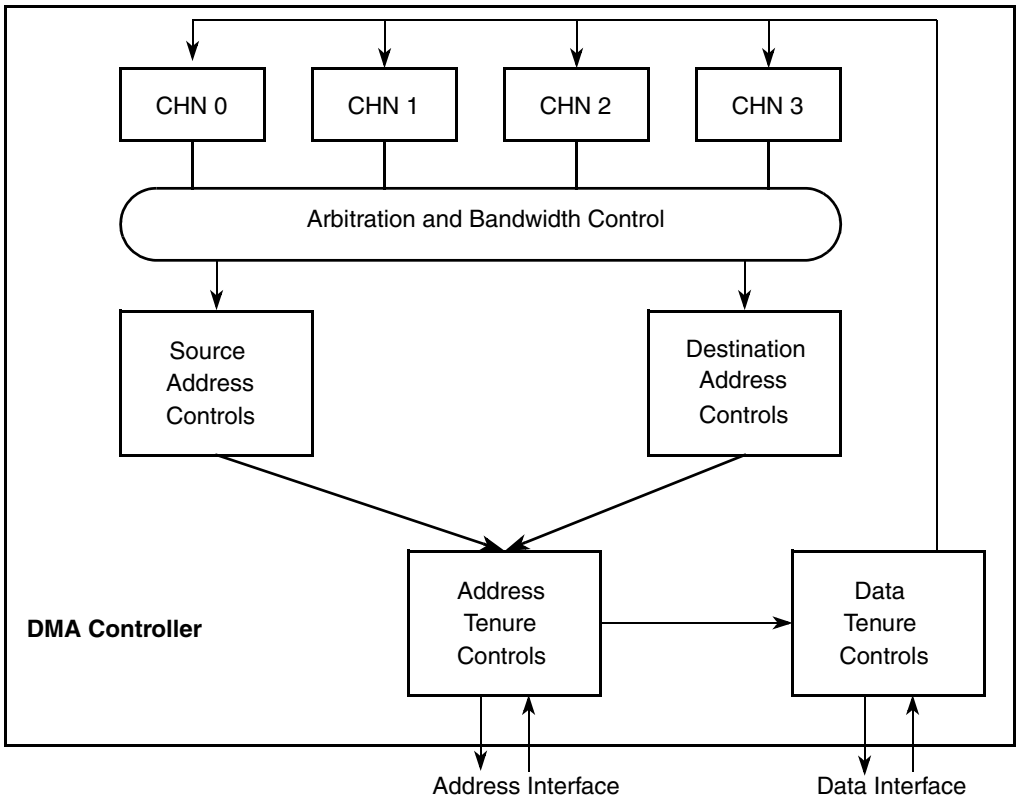


Figure 15-1. DMA Block Diagram

15.1.2 Overview

The DMA controller has four high-speed DMA channels. Both the core and external devices can initiate DMA transfers. All channels are capable of complex data movement and advanced transaction chaining. Figure 15-1 is a high-level block diagram of the DMA controller. Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and write data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

15.1.3 Features

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, and $\overline{\text{DMA\_DDONE}}$

15.1.4 Modes of Operation

The MPC8555E has two modes of operation: basic and extended. Basic mode is the DMA legacy mode. It does not support advanced features. Extended mode supports advanced features like striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. Table 15-1 summarizes the relationship between the modes and the following features:

- Direct mode. No descriptors are involved. Software must initialize the required fields as described in Table 15-1 before starting a transfer.
- Chaining mode. Software must initialize descriptors in memory and the required fields as described in Table 15-1 before starting a transfer.
- Single-write start mode. The DMA process can be started by using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.
- External control capability. This allows an external agent to start, pause, and check the status of a DMA transfer which has already been initialized.

- Channel continue capability. The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- Channel abort capability. The software can abort a previously initiated transfer by setting the bit MR*n*[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 15-1. Relationship of Modes and Features

Mode	Mode with One Additional Feature	Mode with Two Additional Features
B (Basic)	BD (basic direct)	BDS (BD single-write start)
		BDE (BD external control)
	BC (basic chaining)	BCE (BC external control)
		BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
		ExtDE (ExtD external control)
	ExtC (extended chaining)	ExtCE (ExtC external control)
		ExtCS (ExtC single-write start)

Table 15-2 describes bit settings required for each DMA mode of operation.

Table 15-2. DMA Mode Bit Settings

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
Basic Direct Modes					
Basic direct	0	1	0	0	0
Basic direct external control	0	1	0	0	1
Basic direct single-write start	0	1	1	1 or 0	0
Basic Chaining Modes					
Basic chaining	0	0	Reserved	0	0
Basic chaining external control	0	0	Reserved	0	1
Basic chaining single-write start	0	0	Reserved	1	0
Extended Direct Modes					
Extended direct	1	1	0	0	0
Extended direct external control	1	1	0	0	1
Extended direct single-write start	1	1	1	1 or 0	0

Table 15-2. DMA Mode Bit Settings (continued)

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
Extended Chaining Modes					
Extended chaining	1	0	Reserved	0	0
Extended chaining external control	1	0	Reserved	0	1
Extended chaining single-write start	1	0	Reserved	1	0

Refer to Section 15.4, "Functional Description," for details on these modes.

Figure 15-2 shows the general DMA operational flow chart.

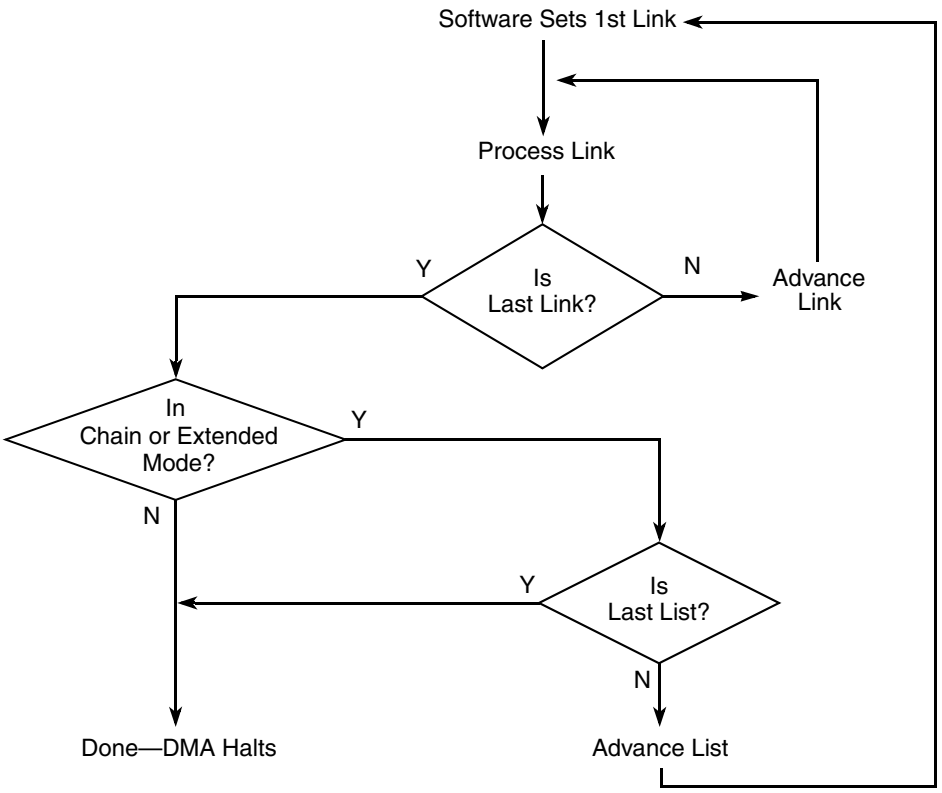


Figure 15-2. DMA Operational Flow Chart

15.2 External Signal Description

This section describes the DMA signals.

15.2.1 Signal Overview

Figure 15-3 summarizes the DMA controller signals.

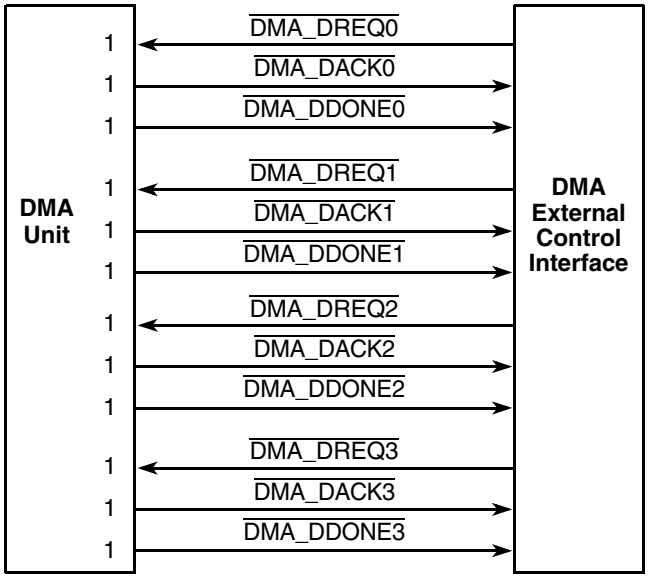


Figure 15-3. DMA Signal Summary

Note that the three DMA signals for DMA channel 3 are multiplexed with the IRQ9–11 signals on the MPC8555E device. These functions are mutually exclusive and the active function is specified in the PMUXCR register of the global utilities block as described in Section 18.4.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

15.2.2 Detailed Signal Descriptions

Table 15-3 describes the DMA signals.

Table 15-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{DMA\_DREQ}n}$ DMA request	I	DMA request. The DMA request signal indicates the start of a DMA transfer or a restart from a paused request. Assertion of $\overline{\text{DMA\_DREQ}n}$ causes MR*n*[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted—Assertion of $\overline{\text{DMA\_DREQ}n}$ while $\overline{\text{DMA\_DACK}n}$ is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion while $\overline{\text{DMA\_DACK}n}$ is asserted results in an illegal condition. Negated—Negation while $\overline{\text{DMA\_DACK}n}$ is asserted has no effect. Negation before the assertion of $\overline{\text{DMA\_DACK}n}$ results in an illegal condition.
		Timing	Assertion—Can be asserted asynchronously Negation— Must remain asserted at least until the assertion of the corresponding $\overline{\text{DMA\_DACK}n}$

Table 15-3. DMA Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{DMA\_DACK}}n$	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress	
		State Meaning	Asserted—Indicates that a DMA transfer is currently in progress. Asserted after the assertion of $\overline{\text{DMA\_DREQ}}n$ to indicate the start of a transfer Negated—Negated after finishing a complete transfer or after entering a paused state if MR*n*[EMP_EN] is set
		Timing	Assertion—Asynchronous assertion; asserted for more than three system clocks Negation—Asynchronous negation; negated for more than three system clocks
$\overline{\text{DMA\_DDONE}}n$	O	DMA done. Indicates that a DMA transfer is complete	
		State Meaning	Asserted—Indicates transfer completion. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface. Negated—Indicates that the current transfer is in process
		Timing	Assertion—Always asserts asynchronously after the negation of the final $\overline{\text{DMA\_DACK}}n$ to indicate completion of a transfer. For a paused transfer, $\overline{\text{DMA\_DDONE}}n$ is asserted asynchronously after the negation of the final $\overline{\text{DMA\_DACK}}n$. Negation—Negated asynchronously after the assertion of $\overline{\text{DMA\_DREQ}}n$ for the next transfer

15.3 Memory Map/Register Definition

This section provides a detailed description of all accessible DMA memory and registers. The descriptions include individual bit level descriptions and reset states of each register. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

15.3.1 Module Memory Map

Table 15-4 lists the DMA registers and their offsets. Note that the full register address is comprised of the programmable CCSRBAR together with the fixed DMA block base address and offset listed in Table 15-4.

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 15-4. DMA Register Summary

Offset	Register	Access	Reset	Section/Page
DMA Controller Block Base Address: 0x2_1000				
0x100	MR0—DMA 0 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x104	SR0—DMA 0 status register	Mixed	0x0000_0000	15.3.2.2/15-11
0x10C	CLNDAR0—DMA 0 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x110	SATR0—DMA 0 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x114	SAR0—DMA 0 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x118	DATR0—DMA 0 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x11C	DAR0—DMA 0 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x120	BCR0—DMA 0 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x128	NLNDAR0—DMA 0 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x134	CLSDAR0—DMA 0 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x13C	NLSDAR0—DMA 0 next list descriptor address register	Mixed	0x0000_0000	15.3.2.11/15-19
0x140	SSR0—DMA 0 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x144	DSR0—DMA 0 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x148–0x17C	Reserved	—	—	—
0x180	MR1—DMA 1 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x184	SR1—DMA 1 status register	Mixed	0x0000_0000	15.3.2.2/15-11
0x18C	CLNDAR1—DMA 1 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x190	SATR1—DMA 1 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x194	SAR1—DMA 1 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x198	DATR1—DMA 1 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x19C	DAR1—DMA 1 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x1A0	BCR1—DMA 1 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x1A8	NLNDAR1—DMA 1 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x1B4	CLSDAR1—DMA 1 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x1BC	NLSDAR1—DMA 1 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x1C0	SSR1—DMA 1 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x1C4	DSR1—DMA 1 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x1C8–0x1FC	Reserved	—	—	—
0x200	MR2—DMA 2 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x204	SR2—DMA 2 status register	Mixed	0x0000_0000	15.3.2.2/15-11
0x20C	CLNDAR2—DMA 2 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x210	SATR2—DMA 2 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14

Table 15-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x214	SAR2—DMA 2 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x218	DATR2—DMA 2 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x21C	DAR2—DMA 2 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x220	BCR2—DMA 2 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x228	NLNDAR2—DMA 2 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x234	CLSDAR2—DMA 2 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x23C	NLSDAR2—DMA 2 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x240	SSR2—DMA 2 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x244	DSR2—DMA 2 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x248–0x27C	Reserved	—	—	—
0x280	MR3—DMA 3 mode register	R/W	0x0000_0000	15.3.2.1/15-9
0x284	SR3—DMA 3 status register	Mixed	0x0000_0000	15.3.2.2/15-11
0x28C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-12
0x290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	15.3.2.4/15-14
0x294	SAR3—DMA 3 source address register	R/W	0x0000_0000	15.3.2.5/15-15
0x298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-16
0x29C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	15.3.2.7/15-16
0x2A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	15.3.2.8/15-17
0x2A8	NLNDAR3—DMA 3 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-17
0x2B4	CLSDAR3—DMA 3 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-18
0x2BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-19
0x2C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	15.3.2.12/15-19
0x2C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	15.3.2.13/15-20
0x2C8–0x2FC	Reserved	—	—	—
0x300	DGSR—DMA general status register	R	0x0000_0000	15.3.2.14/15-21

15.3.2 DMA Register Descriptions

The following sections describe the DMA registers. The majority of these registers are channel-specific and can be identified by one of the four offsets that describe the register.

15.3.2.1 Mode Registers (MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. Figure 15-4 describes the MR*n*.

Offset 0x100, 0x180, 0x200, 0x280 — Access: Read/Write

	0–3	4–7	8	9	10	11–12	13	14	15
R W	—	BWC	—		EMP_EN	—	EMS_EN	DAHTS	
Reset	All zeros								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SAHTS		DAHE	SAHE	—	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM/SWSM	CA	CTM	CC	CS
Reset	All zeros															

Figure 15-4. DMA Mode Registers (MR*n*)

Table 15-5 describes the MR*n* fields.

Table 15-5. MR*n* Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	BWC	Bandwidth/pause control. Defined only if MR*n*[EMP_EN] is set. If multiple channels are executing transfers concurrently the value of MR*n*[BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and switches to the next channel. If only one channel is executing transfers the value of MR*n*[BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of $\overline{\text{DREQ}}$ resumes channel operation. 0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 0100 16 bytes 0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011–1110 Reserved 1111 Disable bandwidth sharing to allow uninterrupted transfers from each channel.
8–9	—	Reserved
10	EMP_EN	External master pause enable. Valid only if MR*n*[EMS_EN] is set. 0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MR*n*[BWC].
11–12	—	Reserved
13	EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MR*n*[CS].

Table 15-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
14–15	DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction while MR*n*[DAHE] is set. The byte count register must be in multiples of the size and the destination address register must be aligned based on the size. The transfer size assigned to MR*n*[DAHTS] must be equal to or smaller than that assigned to MR*n*[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
16–17	SAHTS	Source address hold transfer size. Indicates the transfer size used for each transaction while MR*n*[SAHE] is set. The byte count register must be in multiples of the size and the source address register must be aligned based on the size. The transfer size assigned to MR*n*[SAHTS] must be equal to or smaller than that assigned to MR*n*[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
18	DAHE	Destination address hold enable 0 Disable destination address hold 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MR*n*[DAHTS]. Hardware only supports aligned transfers for this feature.
19	SAHE	Source address hold enable 0 Disable source address hold 1 Enable the DMA controller to hold the source address of a transfer to the size specified by MR*n*[SAHTS]. Hardware only supports aligned transfers for this feature.
20	—	Reserved
21	SRW	Single register write (Direct mode only; reserved for chaining mode.) 0 Normal operation 1 Enable a write to the source address register to simultaneously set MR*n*[CS], starting a DMA transfer, when MR*n*[CDSM/SWSM] is also set. Setting this bit and clearing CDSM/SWSM causes a write to the destination address register to simultaneously set MR*n*[CS], starting a DMA transfer.
22	EOSIE	End-of-segments interrupt enable 0 Do not generate an interrupt at the completion of a data transfer. CLNDAR*n*[EOSIE] overrides this bit on a link descriptor basis. 1 Generate an interrupt at the completion of a data transfer (That is, SR*n*[EOSI] is set). This bit overrides the CLNDAR*n*[EOSIE].
23	EOLNIE	End-of-links interrupt enable 0 Do not generate an interrupt at the completion of a list of DMA transfers. 1 Generate an interrupt at the completion of a list of DMA transfers (That is, NLNDAR*n*[EOLND] is set).
24	EOLSIE	End-of-lists interrupt enable 0 Do not generate an interrupt at the completion of all DMA transfers. 1 Generate an interrupt at the completion of all DMA transfers (That is, NLNDAR*n*[EOLND] and NLSDAR*n*[EOLSD] are set).
25	EIE	Error interrupt enable 0 Do not generate an interrupt if a programming or transfer error is detected. 1 Generate an interrupt if a programming or transfer error is detected.

Table 15-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
26	XFE	Extended features enable 0 Disable the new chaining features. 1 Enable the new chaining features.
27	CDSM/ SWSM	• In chaining mode: current descriptor start mode/single-write start mode • In basic mode (MR*n*[XFE] is cleared), setting this bit causes a write to the current link descriptor address register to simultaneously set MR*n*[CS], starting a DMA transfer. • In extended chaining mode (MR*n*[XFE] is set), setting this bit causes a write to the current list descriptor address register to simultaneously set MR*n*[CS], starting a DMA transfer. • In direct mode: Setting this bit and MR*n*[SRW] causes a write to the source address register to simultaneously set MR*n*[CS], starting a DMA transfer. Clearing this bit and setting MR*n*[SRW] causes a write to the destination address register to simultaneously set MR*n*[CS], starting a DMA transfer. This bit must be cleared when MR*n*[SRW] is cleared.
28	CA	Channel abort 0 No effect 1 Cause the current transfer to be aborted and SR*n*[CB] to be cleared if the channel is busy. The channel remains in the idle state until a new transfer is programmed.
29	CTM	Channel transfer mode 0 Configure the channel in chaining mode. 1 Configure the channel into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the DMA process.
30	CC	Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode. 0 No effect 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31	CS	Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. 0 Halt the DMA process if channel is busy (SR*n*[CB] is set). No effect if the channel is not busy. 1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

15.3.2.2 Status Registers (SR*n*)

The status registers, shown in Figure 15-5, report various DMA conditions during and after a DMA transfer.

Offset 0x104 0x184 0x204 0x284 Access: Mixed

	0	23	24	25	26	27	28	29	30	31
R	—		TE	—	CH	PE	EOLNI	CB	EOSI	EOLSI
W			w1c			w1c	w1c		w1c	w1c
Reset	All zeros									

Figure 15-5. Status Registers (SR*n*)

Table 15-6 describes the bits of the SR*n*.

Table 15-6. SR*n* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer 1 Error condition during the DMA transfer. See Section 15.4.3, “DMA Errors,” for additional information.
25	—	Reserved
26	CH	Channel halted. Cleared automatically by hardware if MR*n*[CS] is set again for resuming a halted transfer 0 Channel is not halted. If software attempts to halt an idle channel (SR*n*[CB] is cleared), this bit will remain 0. 1 DMA transfer was successfully halted by software and can be resumed.
27	PE	Programming error (bit reset, write 1 to clear) 0 No programming error detected 1 A programming error is detected that prevents the DMA transfer from occurring.
28	EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29	CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 A DMA transfer is currently in progress.
30	EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MR*n*[EOSIE] is set or if CLNDAR*n*[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if MRn[EOSIE] is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31	EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

15.3.2.3 Current Link Descriptor Address Registers (CLNDAR*n*)

Current link descriptor address registers contain the address of the current link descriptor. In basic chaining mode, shown in Figure 15-6, software must initialize these registers to point to the first link descriptors in memory.

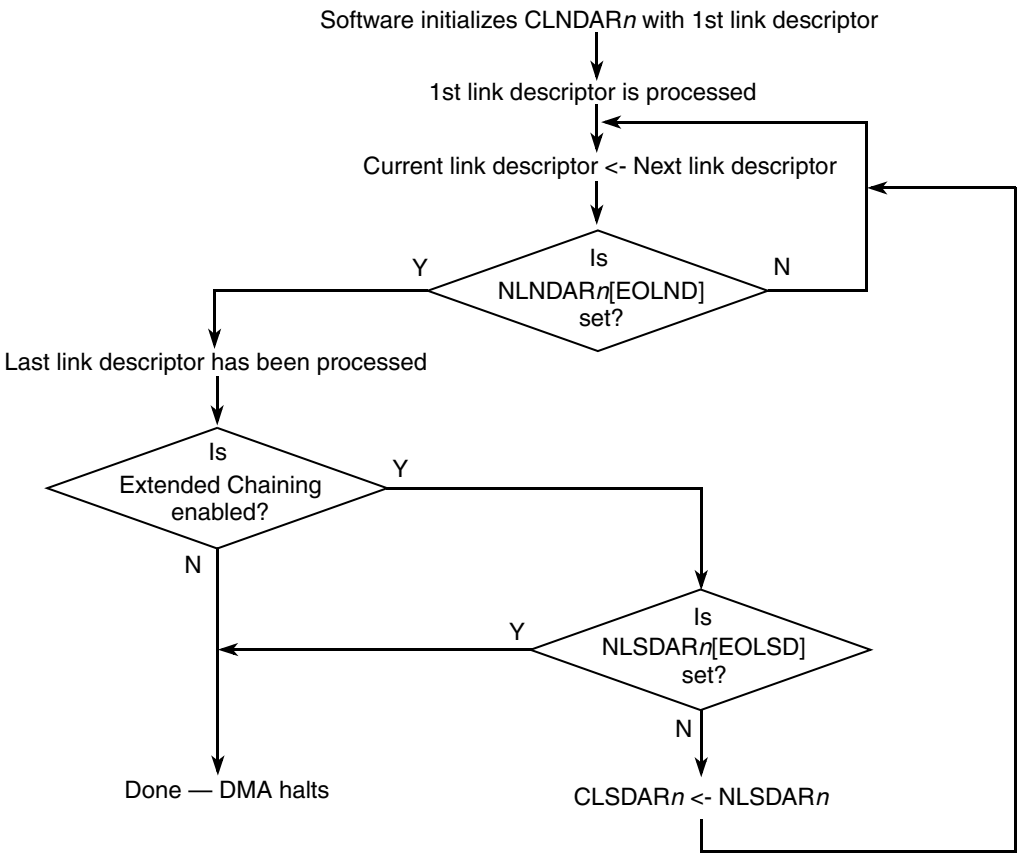


Figure 15-6. Basic Chaining Mode Flow Chart

After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address register and NLNDAR*n*[EOLND] in the next link descriptor address register is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR*n*[EOLSD] in the next list descriptor address register. If EOLSD is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts.

Figure 15-7 shows CLNDAR*n*.

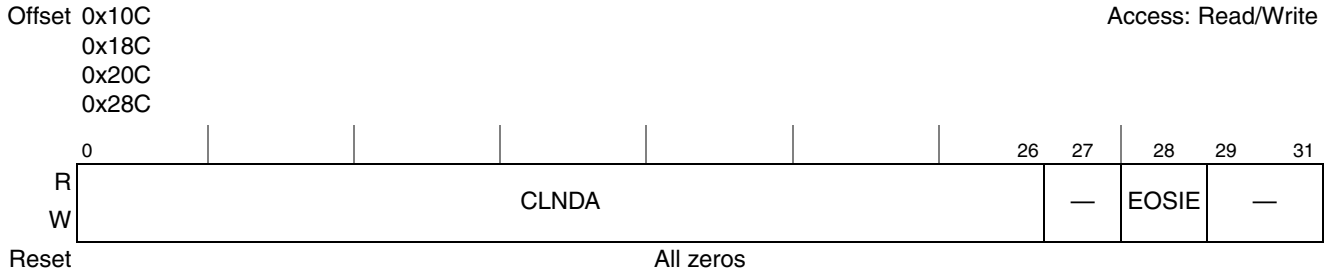


Figure 15-7. Current Link Descriptor Address Registers (CLNDAR*n*)

Table 15-7 describes the fields of the CLNDAR*n*.

Table 15-7. CLNDAR*n* Field Descriptions

Bits	Name	Description
0–26	CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary.
27	—	Reserved
28	EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31	—	Reserved

15.3.2.4 Source Attributes Registers (SATR*n*)

The source attributes registers, shown in Figure 15-8, contain the transaction attributes to be used for the DMA operation. Stride mode is enabled by setting SATR*n*[SSME]. Source read transaction type is specified using SATR*n*[SREADTTYPE].

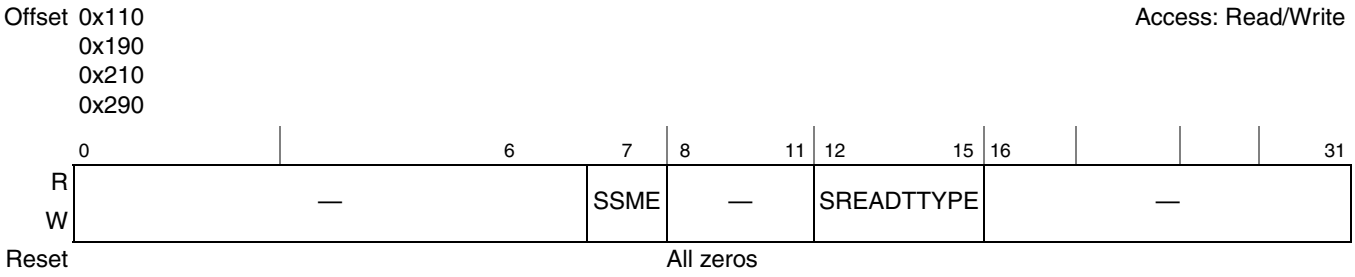


Figure 15-8. Source Attributes Registers (SATR*n*)

Table 15-8 describes the fields of the SATR*n*.

Table 15-8. SATR*n* Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	SSME	Source stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR*n*[XFE] is cleared). Striding on the source address can be accomplished by enabling SATR*n*[SSME] and setting the desired stride size and distance in the SSR*n*.
8–11	—	Reserved
12–15	SREADTTYPE	DMA source transaction type. Reserved values will result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space 0000–0001 Reserved 0011 Reserved 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0111 Read, unlock L2 cache line 1000–1111 Reserved
16–31	—	Reserved

15.3.2.5 Source Address Registers (SAR*n*)

The source address registers, shown in Figure 15-9, contain the address from which the DMA controller reads data. In direct mode, if MR*n*[CDSM/SWSM] and MR*n*[SRW] are set, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

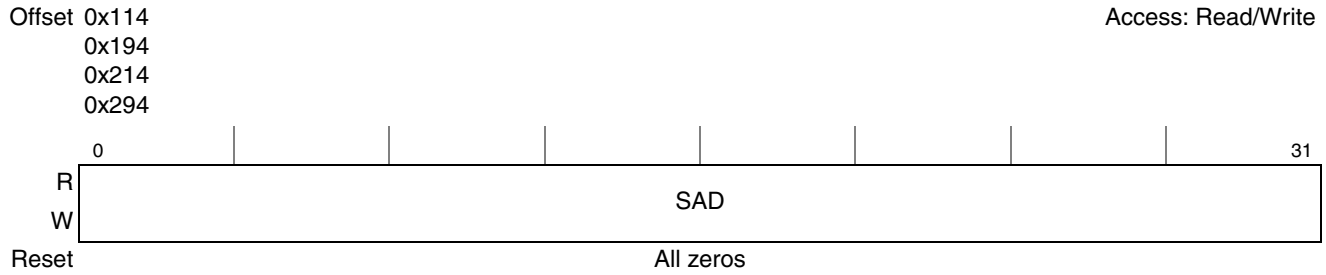


Figure 15-9. Source Address Registers (SAR*n*)

Table 15-9 describes the field of the SAR*n*.

Table 15-9. SAR*n* Field Descriptions

Bits	Name	Description
0–31	SAD	Source address. This register contains the source address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

15.3.2.6 Destination Attributes Registers (DATR*n*)

The destination attributes registers, shown in Figure 15-10, contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DATR*n*[DSME]. Destination write transaction type is specified using the DATR*n*[DWRITETTYPE] field.

The target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DATR*n*[DWRITETTYPE] using the local address space category.

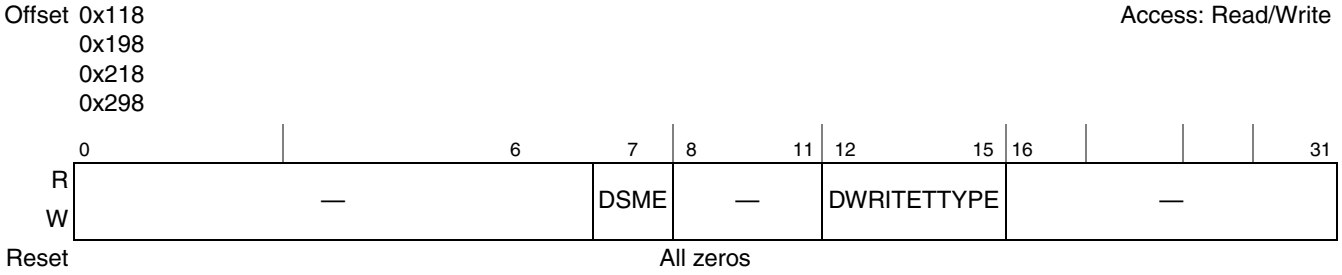


Figure 15-10. Destination Attributes Registers (DATR*n*)

Table 15-10 describes the fields of the DATR*n*.

Table 15-10. DATR*n* Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	DSME	Destination stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR*n*[XFE] is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in DSR*n*.
8–11	—	Reserved
12–15	DWRITETTYPE	DMA destination transaction type. Reserved values will result in a programming error being detected and logged in SR[PE].Transaction type to run on local address space 0000–0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Write, allocate L2 cache line 0111 Write, allocate and lock L2 cache line 1000–1111 Reserved
16–31	—	Reserved

15.3.2.7 Destination Address Registers (DAR*n*)

The destination address registers, shown in Figure 15-11, contain the addresses to which the DMA controller writes data.

In direct mode, if MR*n*[SRW] is set and MR*n*[CDSM/SWSM] is cleared, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

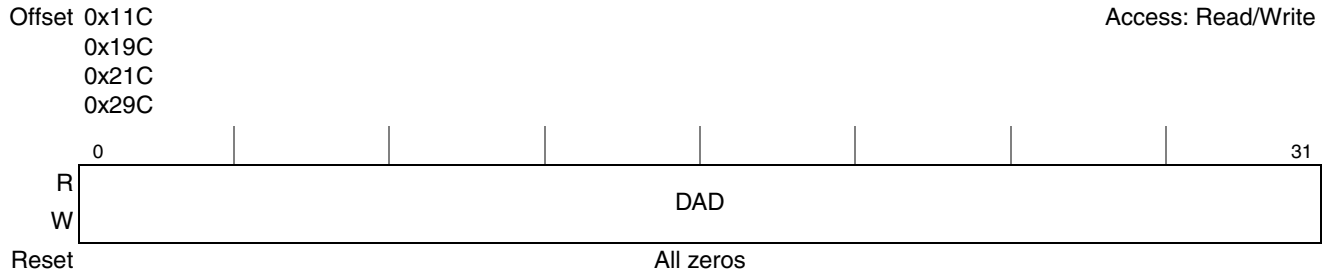


Figure 15-11. Destination Address Registers (DAR*n*)

Table 15-11 describes the field of the DAR*n*.

Table 15-11. DAR*n* Field Descriptions

Bits	Name	Description
0–31	DAD	Destination address. This register contains the destination address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

15.3.2.8 Byte Count Registers (BCR*n*)

The byte count register, shown in Figure 15-12, contains the number of bytes to transfer.

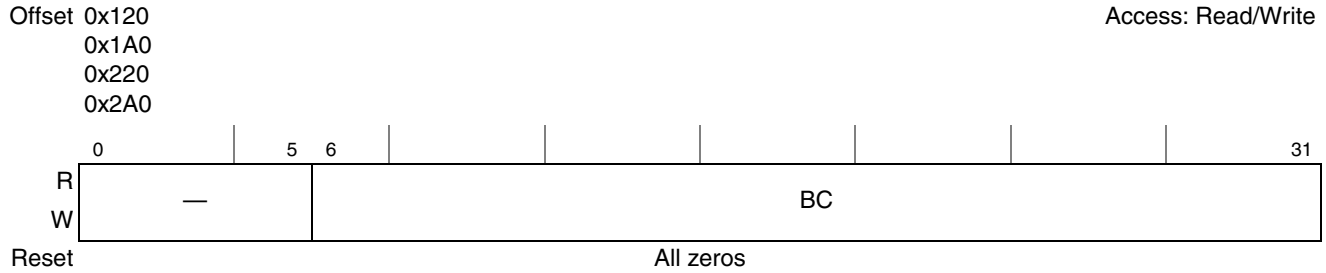


Figure 15-12. Byte Count Registers (BCR*n*)

Table 15-12 describes the fields of the BCR*n*.

Table 15-12. BCR*n* Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6–31	BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is $(2^{26}) – 1$ bytes.

15.3.2.9 Next Link Descriptor Address Registers (NLNDAR*n*)

The next link descriptor address registers, shown in Figure 15-13, contain the address for the next link descriptor in memory. Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

Offset 0x128
0x1A8
0x228
0x2A8

Access: Read/Write

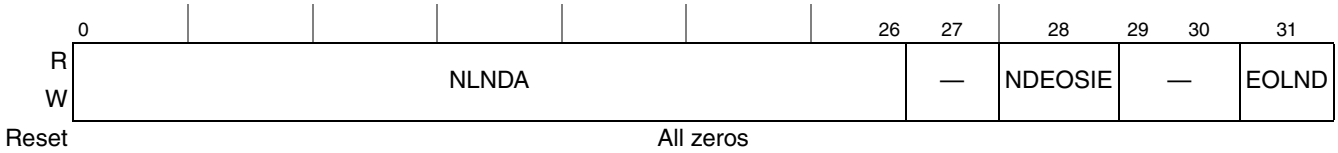


Figure 15-13. Next Link Descriptor Address Registers (NLNDAR*n*)

Table 15-13 describes the fields of the NLNDAR*n* registers.

Table 15-13. NLNDAR*n* Field Descriptions

Bits	Name	Description
0–26	NLNDA	Next link descriptor address. Contains the next link descriptor address in memory. The descriptor must be aligned to a 32-byte boundary.
27	—	Reserved
28	NDEOSIE	Next descriptor end-of-segment interrupt enable 0 Do not generate an interrupt if the current DMA transfer for the current descriptor is finished. 1 Generate an interrupt if the current DMA transfer for the current descriptor is finished.
29–30	—	Reserved
31	EOLND	End-of-links descriptor. This bit is ignored in direct mode. 0 This descriptor is not the last link descriptor in memory for this list. 1 This descriptor is the last link descriptor in memory for this list. If this bit is set, the DMA controller advances to the next list descriptor in memory if NLSDAR*n*[EOLSD] is also set in extended mode.

15.3.2.10 Current List Descriptor Address Registers (CLSDAR*n*)

The current list descriptor address registers, shown in Figure 15-14, contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR*n* to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR*n*[EOLSD] in the next list descriptor address register is clear, the DMA controller reads the new current list descriptor from memory to process that list. If EOLSD in the next list descriptor address register is set and the last link in the current list is finished all DMA transfers are complete.

Offset 0x134
0x1B4
0x234
0x2B4

Access: Read/Write

Figure 15-14. Current List Descriptor Address Registers (CLSDAR*n*)

Table 15-14 describes the fields of the CLSDAR*n*.

Table 15-14. CLSDAR*n* Field Descriptions

Bits	Name	Description
0–26	CLSDA	Current list descriptor address. Contains the current list descriptor address of the buffer descriptor in memory in extended chaining mode. The descriptor must be aligned to a 32-byte boundary.
27–31	—	Reserved

15.3.2.11 Next List Descriptor Address Registers (NLSDAR*n*)

The next list descriptor address registers, shown in Figure 15-15, contain the address for the next list descriptor in memory. If the contents are transferred to the current list descriptor address register they become effective for the current transfer in extended chaining mode.

Table 15-15 describes the fields of the NLSDAR*n*.

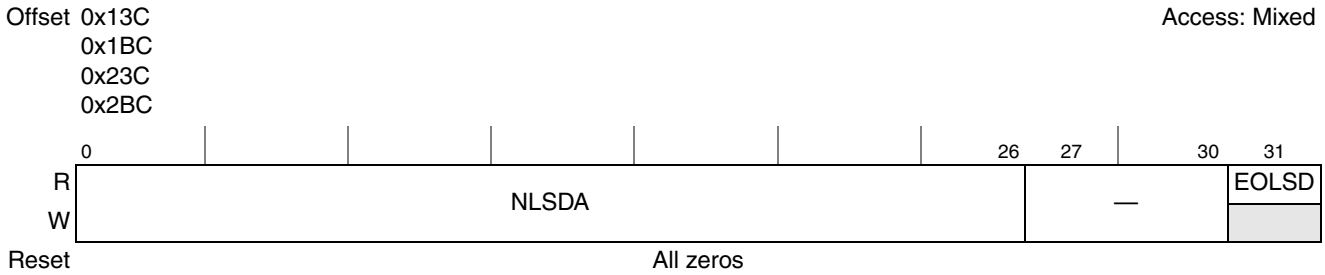


Figure 15-15. Next List Descriptor Address Registers (NLSDAR*n*)

Table 15-15. NLSDAR*n* Field Descriptions

Bits	Name	Description
0–26	NLSDA	Next list descriptor address. Contains the next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27–30	—	Reserved
31	EOLSD	End-of-lists descriptor. This bit is ignored in direct mode. 0 This list descriptor is not the last list descriptor in memory. 1 This list descriptor is the last list descriptor in memory. If this bit is set, then the DMA controller halts after the last link descriptor transaction is finished.

15.3.2.12 Source Stride Registers (SSR*n*)

The source stride register, shown in Figure 15-16, contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.

Offset 0x140
0x1C0
0x240
0x2C0

Access: Read/Write

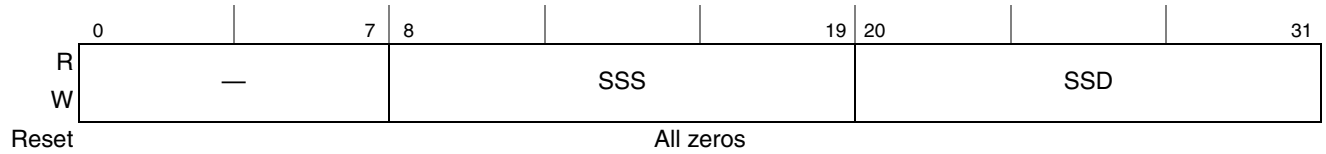


Figure 15-16. Source Stride Registers (SSR*n*)

Table 15-16 describes the fields of the SSR*n*.

Table 15-16. SSR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–19	SSS	Source stride size. Number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
20–31	SSD	Source stride distance. The source stride distance in bytes from start byte to start byte.

15.3.2.13 Destination Stride Registers (DSR*n*)

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated. Figure 15-17 describes the DSR*n*.

Offset 0x144
0x1C4
0x244
0x2C4

Access: Read/Write

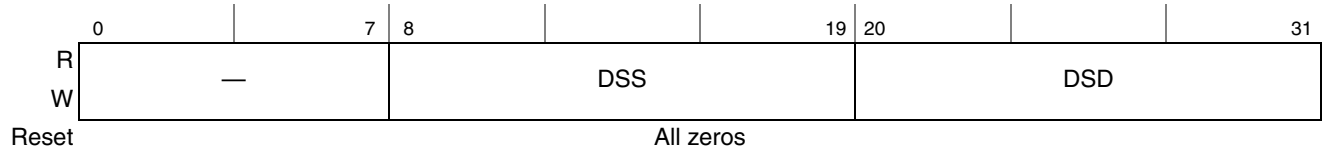


Figure 15-17. Destination Stride Registers (DSR*n*)

Table 15-17 describes the fields of the DSR*n*.

Table 15-17. DSR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–19	DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31	DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

15.3.2.14 DMA General Status Register (DGSR)

The DMA general status register combines all of the status bits from each channel into one register. This register is read-only. Figure 15-18 describes the DGSR.

Offset 0x300 Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TE0	—	CH0	PE0	EOLNI0	CB0	EOSI0	EOLSI0	TE1	—	CH1	PE1	EOLNI1	CB1	EOSI1	EOLSI1
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TE2	—	CH2	PE2	EOLNI2	CB2	EOSI2	EOLSI2	TE3	—	CH3	PE3	EOLNI3	CB3	EOSI3	EOLSI3
W																
Reset	All zeros															

Figure 15-18. DMA General Status Register (DGSR)

Table 15-18 describes the fields of the DGSR.

Table 15-18. DGSR Field Descriptions

Bits	Name	Description
0	TE0	Transfer error, channel 0 0 Normal operation 1 An error condition occurred during the DMA transfer.
1	—	Reserved
2	CH0	Channel halted, channel 0
3	PE0	Programming error, channel 0
4	EOLNI0	End-of-links interrupt, channel 0
5	CB0	Channel busy, channel 0
6	EOSI0	End-of-segment interrupt, channel 0
7	EOLSI0	End-of-lists/direct interrupt, channel 0
8	TE1	Transfer error, channel 1 0 Normal operation 1 An error condition occurred during the DMA transfer.
9	—	Reserved
10	CH1	Channel halted, channel 1
11	PE1	Programming error, channel 1
12	EOLNI1	End-of-links interrupt, channel 1
13	CB1	Channel busy, channel 1
14	EOSI1	End-of-segment interrupt, channel 1
15	EOLSI1	End-of-lists/direct interrupt, channel 1

Table 15-18. DGSR Field Descriptions (continued)

Bits	Name	Description
16	TE2	Transfer error, channel 2 0 Normal operation 1 An error condition occurred during the DMA transfer.
17	—	Reserved
18	CH2	Channel halted, channel 2
19	PE2	Programming error, channel 2
20	EOLNI2	End-of-links interrupt, channel 2
21	CB2	Channel busy, channel 2
22	EOSI2	End-of-segment interrupt, channel 2
23	EOLSI2	End-of-lists/direct interrupt, channel 2
24	TE3	Transfer error, channel 3 0 Normal operation 1 An error condition occurred during the DMA transfer.
25	—	Reserved
26	CH3	Channel halted, channel 3
27	PE3	Programming error, channel 3
28	EOLNI3	End-of-links interrupt, channel 3
29	CB3	Channel busy, channel 3
30	EOSI3	End-of-segment interrupt, channel 3
31	EOLSI3	End-of-lists/direct interrupt, channel 3

15.4 Functional Description

This section describes the function of the DMA controller.

15.4.1 DMA Channel Operation

All DMA channels support two different modes of operation: a basic mode (MR*n*[XFE] is cleared) and an extended mode (MR*n*[XFE] is set). In both modes, a channel can be activated by clearing and setting MR*n*[CS], or through the single-write start mode using MR*n*[CDSM/SWSM] and MR*n*[SRW], or through an external control mode using MR*n*[ECS_EN].

In basic mode, the channel can be programmed in basic direct mode or basic chaining mode. In extended mode, the channel can be programmed in extended direct mode or extended chaining mode. Extended mode provides more capabilities, such as extended descriptor chaining, striding capabilities, and a more flexible descriptor structure.

The DMA controller supports misaligned transfers for both the source and destination addresses. In order to maximize performance, the source and destination engines align the source and destination addresses to a 64-byte boundary. The DMA always reads/writes the maximum number of bytes for a given transfer as

described by the capability inputs of the DMA controller except for globally coherent transactions that use the size of the cache coherence granule as described by the mode select input.

The DMA controller supports bandwidth control, which prevents a channel from consuming all the data bandwidth in the controller. Each channel is allowed to consume the bandwidth of the shared resources as specified by the bandwidth control value. After the channel uses its allotted bandwidth, the arbiter grants the next channel access to the shared resources. The arbitration is round robin between the channels. This feature is also used to implement the external control pause feature. If the external control start and pause are enabled in the MR*n*, the channel enters a paused state after transferring the data described in the bandwidth control. External control can restart the channel from a paused state.

The DMA programming model permits software to program each DMA engine independently to interrupt on completed segment, chain, or error. It also provides the capability for software to resume the DMA engine from a hardware halted condition by setting the channel continue bit, MR*n*[CC]. See Table 15-19 for more complete descriptions of the channel states and state transitions.

15.4.1.1 Basic DMA Mode Transfer

This mode is primarily included for backward compatibility with existing DMA controllers which use a simple programming model. This is the default mode out of reset. The different modes of operation under the basic mode are explained in the following sections.

15.4.1.1.1 Basic Direct Mode

In basic direct mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing SAR*n*, SATR*n*, DAR*n*, DATR*n*, and BCR*n* registers. The DMA transfer is started when MR*n*[CS] is set. Software is expected to program all the appropriate registers before setting MR*n*[CS] to a 1. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in basic direct mode is as follows:

1. Poll the channel state (see Table 15-19), to confirm that the specific DMA channel is idle.
2. Initialize SAR*n*, SATR*n*, DAR*n*, DATR*n* and BCR*n*.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

15.4.1.1.2 Basic Direct Single-Write Start Mode

In basic direct single-write start mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing the SATR*n*, DATR*n*, and BCR*n* registers. Setting MR*n*[SRW] configures the

DMA controller to begin the DMA transfer either when SAR*n* is written or when DAR*n* is written, determined by the state of MR*n*[CDSM/SWSM]. Writing to SAR*n* initiates the DMA transfer if MR*n*[CDSM/SWSM] is set. Writing to DAR*n* initiates the DMA transfer if MR*n*[CDSM/SWSM] is cleared. The DMA controller automatically sets the channel start bit, MR*n*[CS]. Software is expected to program all the appropriate registers before writing the source or destination address registers. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in single-write start basic direct mode is as follows:

1. Poll the channel state (see Table 15-19), to confirm that the specific DMA channel is idle.
2. Initialize the source attributes (SATR*n*), DATR*n*, and BCR*n* registers.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], and the single-write start direct mode bit, MR*n*[SRW]. Other control parameters may also be initialized in the mode register. Set MR*n*[CDSM/SWSM] for transfers started using SAR*n*. Clear MR*n*[CDSM/SWSM] for transfers started using the DAR*n*.
4. A write to the source or destination address register starts the DMA transfer and automatically sets MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

15.4.1.1.3 Basic Chaining Mode

In basic chaining mode, software must first build link descriptor segments in memory. Then the current link descriptor address register must be initialized to point to the first descriptor in memory. The DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded for the segment. After the current segment is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. The transfer is finished if the current link descriptor is the last one in memory or if an error condition occurs. The sequence of events to start and complete a transfer in chaining mode is as follows:

1. Build link descriptor segments in memory.
2. Poll the channel state (see Table 15-19), to confirm that the specific DMA channel is idle.
3. Initialize CLNDAR*n* to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], as well as MR*n*[XFE], to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

15.4.1.1.4 Basic Chaining Single-Write Start Mode

Basic chaining single-write start mode allows a chain to be started by writing the current link descriptor address register (CLNDAR*n*). Setting MR*n*[CDSM/SWSM] in the mode register causes MR*n*[CS] to be automatically set when the current link descriptor address register is written. The sequence of events to start and complete a chain using single-write start mode is as follows:

1. Set the mode register current descriptor start mode bit, MR*n*[CDSM/SWSM], and the extended features enable bit MR*n*[XFE]. Also, clear the channel transfer mode bit, MR*n*[CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the channel state (see Table 15-19), to confirm that the specific DMA channel is idle.
4. Initialize CLNDAR*n* to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

15.4.1.2 Extended DMA Mode Transfer

The extended DMA mode also operates in chaining and direct mode. It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting MR*n*[XFE].

15.4.1.2.1 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

15.4.1.2.2 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

15.4.1.2.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then CLSDAR*n* must be initialized to point to the first list descriptor in memory. The DMA controller loads list

descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see Table 15-19), to confirm that the specific DMA channel is idle.
3. Initialize CLSDAR*n* to point to the first list descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], to indicate chaining mode. MR*n*[XFE] must be set to indicate extended DMA mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

15.4.1.2.4 Extended Chaining Single-Write Start Mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer. Setting MR*n*[CDSM/SWSM] causes MR*n*[CS] to be set automatically when CLSDAR*n* is written. The sequence of events to start and complete an extended chain using single-write start mode is as follows:

1. Set MR*n*[CDSM/SWSM], MR*n*[CTM], and MR*n*[XFE] to indicate extended chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build list and link descriptor segments in local memory.
3. Poll the channel state (see Table 15-19), to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

15.4.1.3 External Control Mode Transfer

An external control can be used to control all DMA channels by setting MR*n*[EMS_EN]. The external control can control the DMA channel in the following transfer modes:

- Basic direct

- Basic chaining
- Extended direct
- Extended chaining

Note that when operating the DMA in chaining mode, the register byte count field, BCR[BC], must be initialized to zero before enabling the pause feature. In chaining modes, the channel does not pause for descriptor fetch transfer.

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or restart a paused DMA transfer. The DMA controller acknowledges a DMA transfer in progress and also indicates a transfer completion.

The pause feature can be enabled by setting MR*n*[EMP_EN]. MR*n*[BWC] specifies how much data to allow a specific channel to transfer before entering a paused state by clearing MR*n*[CS]. Note, however, that write data for a paused transfer may not have reached the target interface when so indicated. The channel can be restarted from a paused state by the asserted edge of $\overline{\text{DREQ}}$ as driven by an external master. In chaining modes, the channel does not pause for descriptor fetch transfer. It only pauses during the actual data transfer.

The following signals are defined for the external control interface:

- $\overline{\text{DMA\_DREQ}}$ - Asserting edge triggers a DMA transfer start or restart from a pause request. Sets MR*n*[CS].
- $\overline{\text{DMA\_DACK}}$ - Indicates a DMA transfer currently in progress. SR*n*[CB] is set.
- $\overline{\text{DMA\_DDONE}}$ - Indicates the completion of the DMA controller's involvement in the transfer and the readiness to accept a new DMA command. SR*n*[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

Detailed descriptions of the external control interface are in Table 15-3. The timing diagram of the external control interface is shown in Figure 15-19.

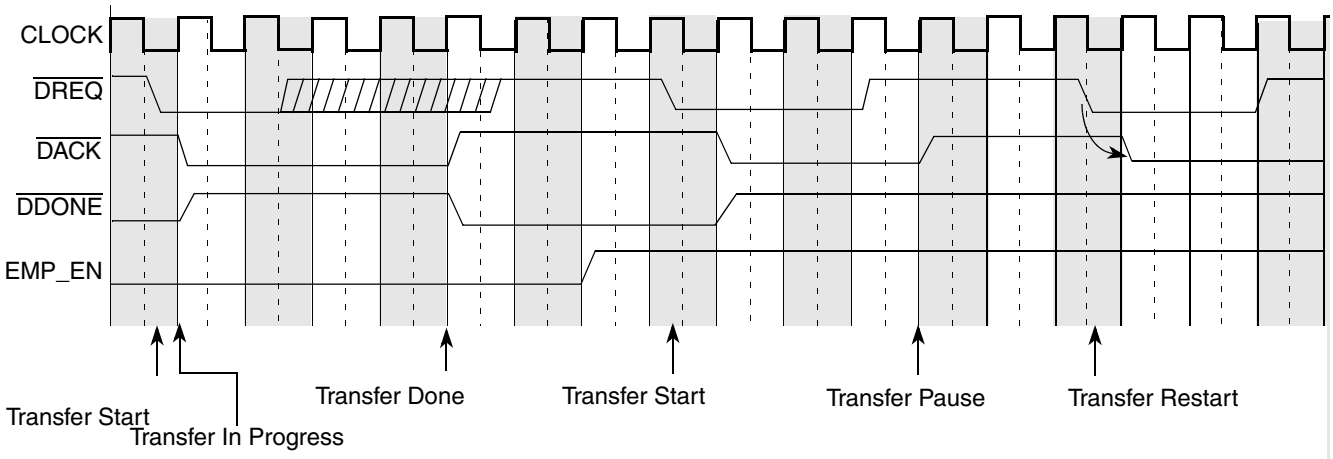


Figure 15-19. External Control Interface Timing

15.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode (enabled when MR*n*[CC] is set) offers software the flexibility of having the DMA controller get started on descriptors that have already been programmed while software continues to build more descriptors in memory. Software can set the end-of-links descriptor (EOLND) in basic mode, or end-of-lists descriptor (EOLSD) in extended mode, to cause the channel to go into a halted state while software continues to build other descriptors in memory. Software can then set CC to force hardware to continue where it left off. channel continue is only meaningful for chaining modes, not direct mode.

If CC is set by software while the channel is busy with a transfer, the DMA controller finishes all transfers until it reaches the EOLND in basic mode or EOLSD in extended mode. The DMA controller then refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If EOLND or EOLSD is not set, the DMA controller continues the transfer by refetching the new descriptor.

If CC is set by software while the channel is not busy with a transfer, the DMA controller refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If the EOLND or EOLSD bits are not set, the DMA controller continues the transfer by refetching the new descriptor.

15.4.1.4.1 Basic Mode

On a channel continue, the descriptor at the current link descriptor address register (CLNDAR*n*) is refetched to get the next link descriptor address field as updated by software. The channel halts if NLNDAR*n*[EOLND] is still set. If EOLND is zero, the next link descriptor address is copied into CLNDAR*n* and the channel continues with another descriptor fetch of the current link descriptor address. As a result, two link descriptor fetches always exist after channel continue before starting the first transfer.

15.4.1.4.2 Extended Mode

On a channel continue, the descriptor at the current list descriptor (CLSDAR*n*) address register is refetched to get the next list descriptor address field as updated by software. The channel halts if NLSDAR*n*[EOLSD] is still set. If not, the next list descriptor address is copied into the CLSDAR*n* register and the channel continues with another descriptor fetch of the current list descriptor address. As a result, two list descriptor fetches always exist after channel continue before the first link descriptor fetch and the first transfer.

15.4.1.5 Channel Abort

Software can abort a previously initiated transfer by setting MR*n*[CA]. Once the DMA channel controller detects a zero-to-one transition of MR*n*[CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR*n*[CB]. Successful completion of a software initiated abort request can be recognized by MR*n*[CA] being set and SR*n*[CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR*n*[TE] is set), or the channel has completed all transfers, then SR*n*[CB] being cleared may not signify that the controller entered a halt state due to the abort request.

15.4.1.6 Bandwidth Control

MR*n*[BWC] specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware. This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 Kbyte at a time when no other channel is active.

15.4.1.7 Channel State

Table 15-19 defines the state of a channel based on the values of the channel start (MR*n*[CS]), channel busy (SR*n*[CB]), transfer error (SR*n*[TE]), and channel continue (MR*n*[CC]) bits.

Table 15-19. Channel State Table

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[CC]	Channel State
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel Continue unexpected. Channel remains in error halt state
0	1	0	0	Software halted channel. The channel was busy and software cleared MR*n*[CS].
0	1	0	1	Channel remains in halt state.
—	1	1	—	The channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In direct mode, the channel continue has no effect.
1	0	1	0	Error occurred during transfer
1	0	1	1	Channel remains in error halt state
1	1	0	0	Transfer in progress
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue

15.4.1.8 Illustration of Stride Size and Stride Distance

If operating in stride mode, the stride size defines the amount of data to transfer before jumping to the next quantity of data as specified by the stride distance. The stride distance is added to the current base address to point to the next quantity of data to be transferred. Figure 15-20 illustrates the stride size and distance parameters. As shown, each time the stride distance is added to the base address, the resulting address becomes the new base address. This sequence repeats until the amount of data transferred equals the transfer size.

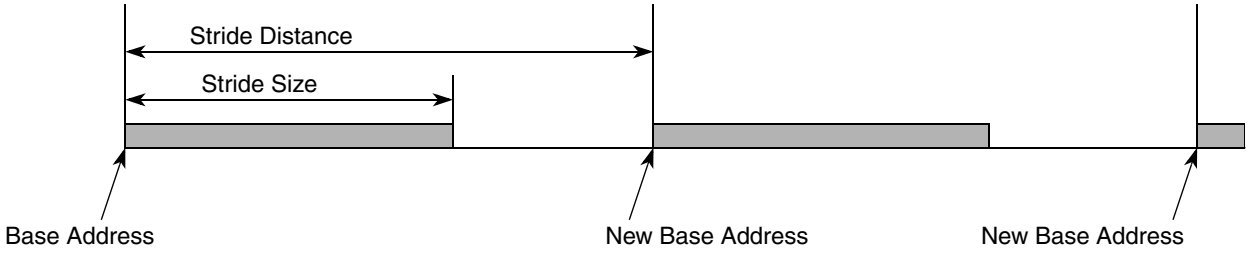


Figure 15-20. Stride Size and Stride Distance

15.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

15.4.3 DMA Errors

On a transfer error (uncorrectable ECC errors on memory accesses, parity errors on local bus or PCI, address mapping errors, for example), the DMA halts by setting SR*n*[TE] and generates an interrupt if MR*n*[EIE] is set. On a programming error, the DMA sets SR*n*[PE] and generates an interrupt if MR*n*[EIE] is set. The DMA controller detects the following programming errors:

- Transfer started with a byte count of zero
- Stride transfer started with a stride size of zero
- Transfer started with a priority of three
- Illegal type, defined by SATR*n*[SREADTTYPE] and DATR*n*[DWRITETTYPE], used for the transfer.

15.4.4 DMA Descriptors

The DMA engine recognizes list descriptors and link descriptors. List descriptors connect lists of link descriptors. Link descriptors describe the DMA activity that is to take place. DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the EOLND bit in the next link descriptor; the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Link and list descriptor fetches always snoop the local memory space.

NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

The last link descriptor in the last list in memory sets NLNDAR*n*[EOLND] in the next link descriptor and NLSDAR*n*[EOLSD] in the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met

as shown in Figure 15-21. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor.

Table 15-20 summarizes the DMA list descriptors.

Table 15-20. List DMA Descriptor Summary

Descriptor Field	Description
Next list descriptor address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor address registers.
First link descriptor address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list

Table 15-21 summarizes the DMA link descriptors.

Table 15-21. Link DMA Descriptor Summary

Descriptor Field	Description
Source attributes register	Contains source transaction attributes
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the Source address register.
Destination attributes register	Contains destination transaction attributes
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the destination address register.
Next link descriptor address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the next link descriptor address registers.
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the byte count register.

Figure 15-21 describes the DMA transaction flow.

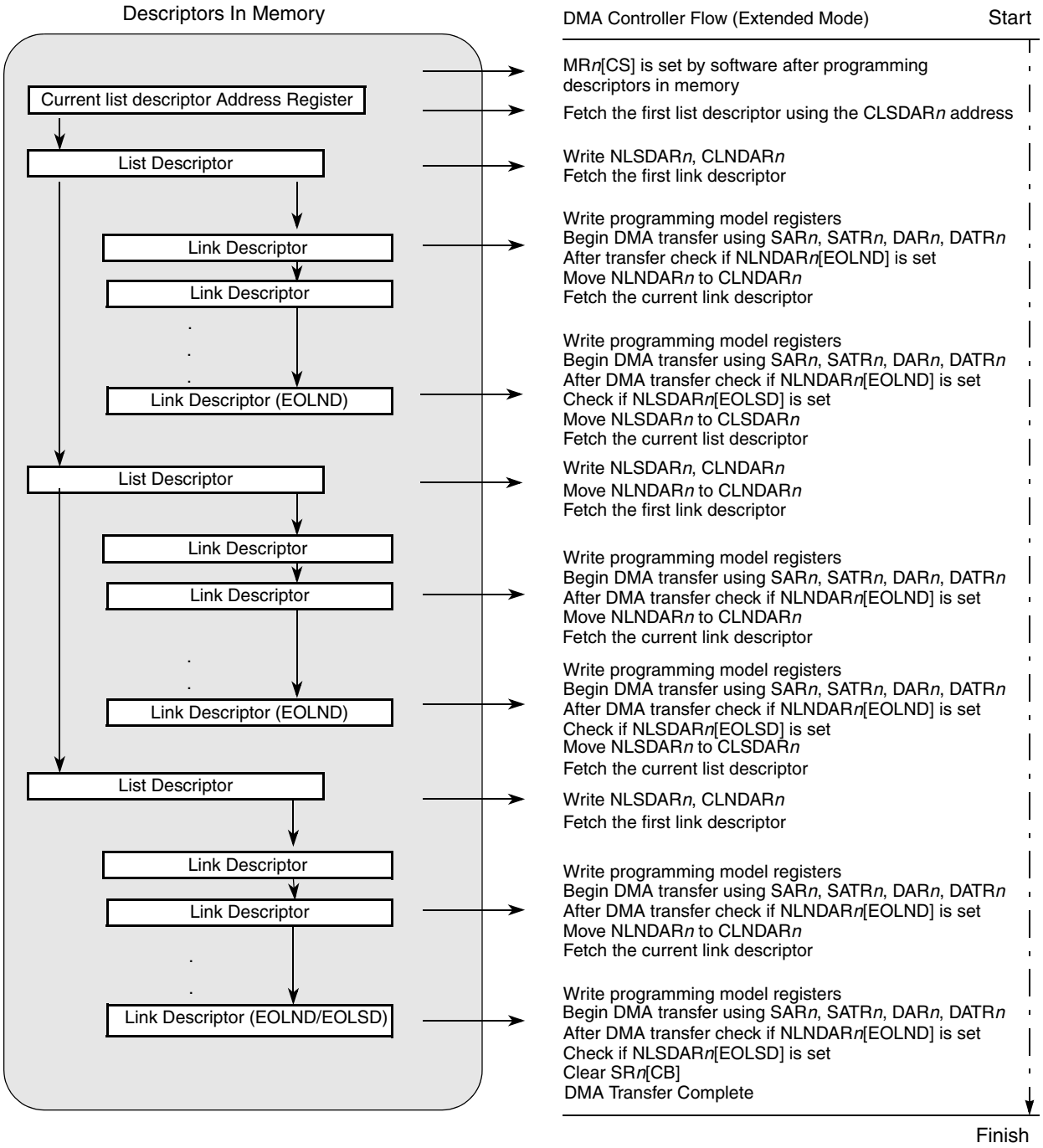


Figure 15-21. DMA Transaction Flow with DMA Descriptors

Figure 15-22 describes the format of the list descriptors.

Offset	
0x00	Reserved
0x04	Next List Descriptor Address
0x08	Reserved
0x0C	First Link Descriptor Address
0x10	Source Stride
0x14	Destination Stride
0x18	Reserved
0x1C	Reserved

Figure 15-22. List Descriptor Format

Figure 15-23 describes the format of the link descriptors.

Offset	
0x00	Source Attributes
0x04	Source Address
0x08	Destination Attributes
0x0C	Destination Address
0x10	Reserved
0x14	Next Link Descriptor Address
0x18	Byte Count
0x1C	Reserved

Figure 15-23. Link Descriptor Format

15.4.5 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the DMA controller and is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the DMA controller are the following:

- Due to the limited number of buffers that the DMA controller can use, stride sizes less than 64 bytes should be avoided. Maximum utilization is obtained from strides greater than or equal to 256 bytes. However, small stride sizes can be used for scatter-gather functions.
- Coherent reads or writes are broken up into cache line accesses in the DMA.

The DMA controller restrictions are as follows:

- All interface capabilities from where descriptors are being fetched must support read sizes of 32 bytes or greater.
- If MR*n*[SAHE] is set, the source interface transfer size capability must be greater than or equal to MR*n*[SAHTS]. The source address must be aligned to a size specified by SAHTS.

- If MR*n*[DAHE] is set, the destination interface transfer size capability must be greater than or equal to MR*n*[DAHTS]. The destination address must be aligned to the size specified by DAHTS.
- Destination striding is not supported if MR*n*[DAHE] is set and source striding is not supported if MR*n*[SAHE] is set.

15.5 DMA System Considerations

Figure 15-24 shows the most important data paths within the MPC8555E. Many of these paths are served by captive DMA controllers and virtually all can be served by the general purpose four-channel DMA controller. This section provides information about how to make most effective use of these DMA channels including the following topics:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources and destinations
- Transparency of the bus interfaces to DMA operations
- What is useful as opposed to what is possible. For example, it is possible to address any internal register through the internal control bus, which means configuration and control registers can be DMA source or destination targets. However, the typical use of DMA functionality is to reduce host processor loading by moving large amounts of data with minimal CPU involvement. Using a general-purpose DMA controller to load small amounts of configuration data only makes sense in special circumstances (perhaps during system boot, for example).

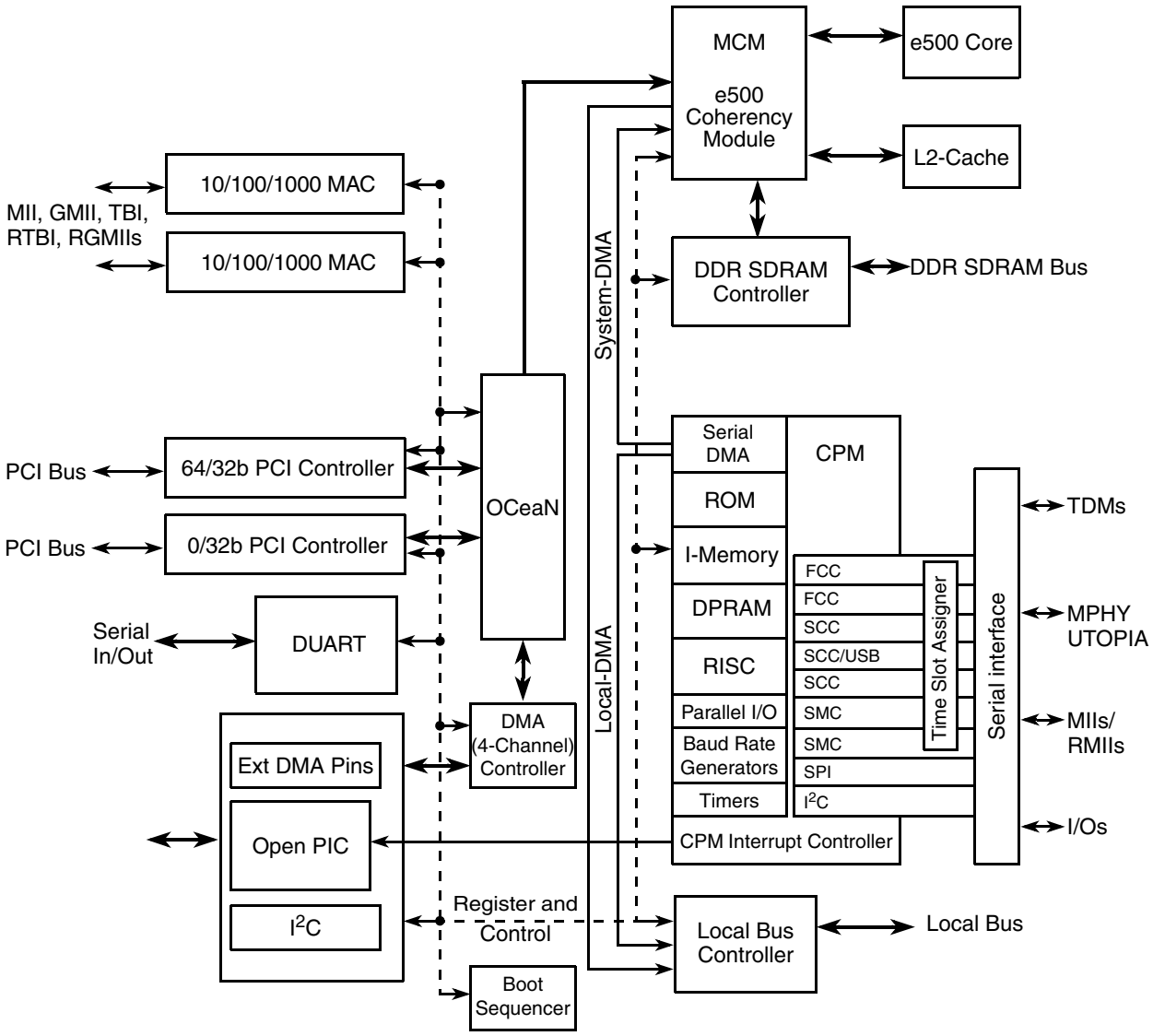


Figure 15-24. DMA Data Paths

Table 15-22 lists all of the DMA controllers (both captive and general-purpose) on the device and the most likely DMA targets on and off-chip. The bus controllers themselves cannot initiate DMA transfers; rather, they operate transparently with respect to both on- and off-chip DMA controllers. The codes in the table cells have the following meaning:

- Y—Supported
- NR—Possible, but not recommended. Inefficient use of system resources
- NS—Possible, but not supported. Resulting system behavior is not defined

Table 15-22. MPC8555E DMA Paths

DMA Controllers		On-Chip Targets			Off-Chip Targets				
		L2	CPM DPRAM	Configuration Registers	DDR	Local Bus	PCI	DUART FIFO	Ethernet Buffers
On-chip	CPM	Y	NR	NS	Y	Y	Y	Y	Y
	Ethernet	Y	NS	NS	Y	Y	Y	Y	Y
	4-channel	Y	Y	NR	Y	Y	Y	Y	NS
Off-chip	PCI controller	Y	Y	NS	Y	Y	—	Y	Y

Note: On-chip target configuration registers include I^2C data register.
On-chip CPM and Ethernet are captive resources. Not available to external masters.
On-chip 4-channel controller can serve external masters.

15.5.1 Unusual DMA Scenarios

The following is a description of unusual DMA paths including explanations of why some functional blocks cannot serve as DMA targets. The following topics are addressed:

- Transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through an internal control bus, which means configuration and control registers can be DMA targets.

15.5.1.1 DMA to Core

The L1 cache cannot be a direct DMA target because it cannot be directly addressed by software. However, DMA access into the L1 cache occurs indirectly if a block of memory that is cached in the L1 is specified as the DMA target. This effect is deterministic if the target memory block was locked into the L1 with cache locking instructions.

15.5.1.2 DMA to CPM

The CPM's dual port RAM (DPRAM) can serve as both a source and destination for general-purpose DMA transfers. However, because the CPM has its own dedicated DMA controller, using an external DMA controller to access the DPRAM makes little sense except in special circumstances (such as, possibly, during system initialization).

15.5.1.3 DMA to Ethernet

The Ethernet controllers cannot serve as DMA targets because they have no suitable internal memory for this purpose. The Ethernet controllers have dedicated DMA channels to move data between the external transmit and receive buffers and the internal packet buffer. This dedicated channel is the only DMA service to the internal packet buffers.

However, Ethernet ports can serve as DMA targets by using a general-purpose DMA controller to access the transmit and receive buffers defined by the Ethernet buffer descriptors. Because Ethernet data buffers are located in RAM outside of the Ethernet controllers, general-purpose DMA engines can move data to or from these memory regions. Also, because Ethernet controllers automatically read buffer descriptors and send (or load) data buffers, a DMA transfer into (or out of) these buffers is effectively a transfer into (or out of) the Ethernet ports.

15.5.1.4 DMA to Configuration and Control Registers

Because any internal register can be addressed with the four-channel DMA controller, configuration and control registers throughout the device are valid DMA targets. However, the primary purpose of DMA—to reduce processor load by moving large blocks of data— is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the I^2C controller or programmable interrupt controller (PIC), doing so is extremely inefficient and is seldom beneficial in normal operation. The overhead of creating DMA descriptors far exceeds any savings in CPU cycles.

15.5.1.5 DMA to I^2C

The I^2C controller is not transparent to DMA transfers. Observe the caveats listed in Section 15.5.1.4, “DMA to Configuration and Control Registers,” when accessing any I^2C register, including the data register (I2CDR).

15.5.1.6 DMA to DUART

The DUART provides complete and sophisticated DMA support which is described in Chapter 12, “DUART,” specifically, Section 12.4.5, “FIFO Mode.”

Chapter 16
PCI Bus Interface

The MPC8555E PCI interface complies with the *PCI Local Bus Specification*, Rev. 2.2. It is beyond the scope of this manual to document the intricacies of the PCI. This chapter describes the PCI controller of this device and provides a basic description of the PCI bus operations. The specific emphasis is directed at how the MPC8555E implements the PCI. Designers of systems incorporating PCI devices should refer to the respective specification for a thorough description of the PCI buses.

NOTE

Much of the available PCI literature refers to a 16-bit quantity as a word and a 32-bit quantity as a dword. Note that this is inconsistent with the terminology in the rest of this manual where the terms ‘word’ and ‘double word’ refer to a 32-bit and 64-bit quantity, respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

16.1 Introduction

The PCI controller acts as a bridge between the PCI interface and the OCeaN switch fabric. Figure 16-1 is a high-level block diagram of the PCI controller.

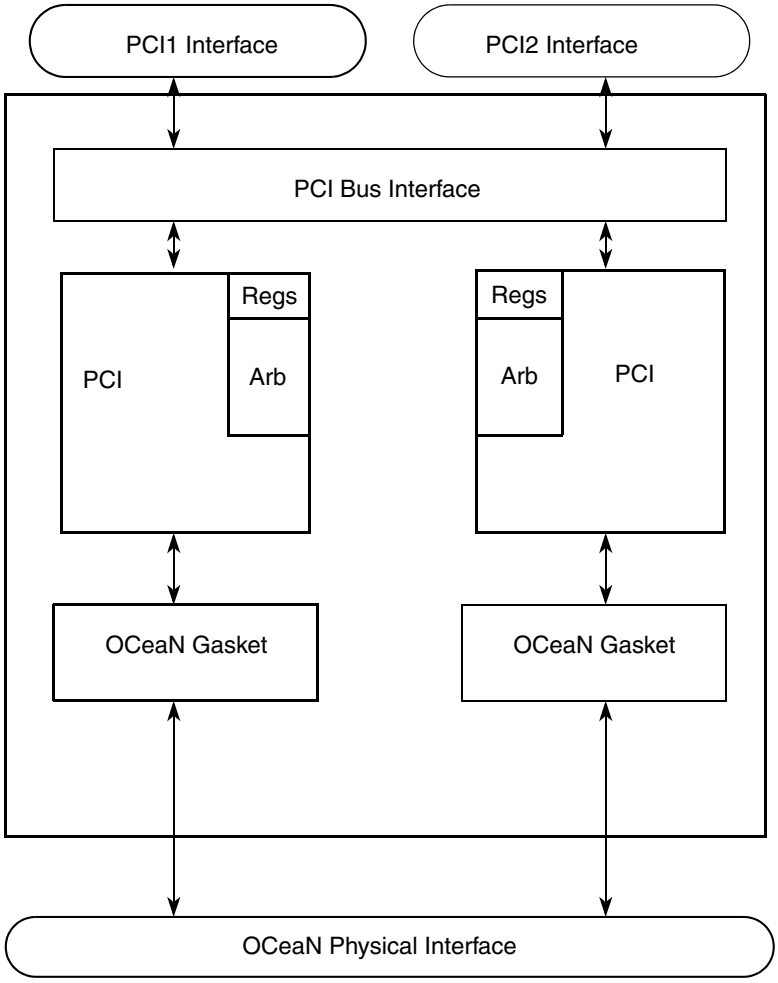


Figure 16-1. PCI Controller Block Diagram

16.1.1 Overview

The PCI controller connects the OCeaN to the PCI bus, to which I/O components are connected. The PCI bus uses a 32- or 64-bit multiplexed address/data bus, plus various control and error signals. The PCI interface supports address and data parity with error checking and reporting.

The PCI interface of the MPC8555E functions both as a master (initiator) and a target device. Internally, the design is divided into the following:

- Data path blocks
- Control logic blocks
- Memory

The data path blocks contain the queues, tables for transaction tracking, and ordering. The control blocks contain control logic and state-machines for buffer control, bus protocol, tag generation, and transaction resizing. The memory blocks are used solely for inbound and outbound data storage.

This allows the MPC8555E to handle separate PCI transactions simultaneously. For example, consider the case where a burst-write transaction from the MPC8555E to another PCI device terminates with a disconnect before finishing the transaction. If another PCI device is granted the PCI bus and requests a burst-read from local memory, the MPC8555E, as a target, can accept the burst-read transfer. When the MPC8555E is granted mastership of the PCI bus, the burst-write transaction continues.

There are two blocks of memory in the design:

- The inbound buffers
- The outbound read buffers combined with the outbound write buffers

There are many blocks of control logic in the block. On the PCI side there are machines for PCI controller initiated address and data tenures for inbound and outbound data, respectively. On the OCeaN side there are machines for fabric arbitration, outbound data, and inbound data.

As an initiator, the MPC8555E supports read and write operations to the PCI memory space, the PCI I/O space, and the 256-byte PCI configuration header space. As an initiator, the MPC8555E also supports generating PCI special-cycle and interrupt-acknowledge transactions. As a target, the MPC8555E supports read and write operations to local memory, and, when configured in agent mode, read and write operations to the internal PCI configuration registers.

The PCI1 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 16.1.3.1.1, “Host Mode,” for more information.The PCI2 interface only supports host mode.

In agent mode, all of the PCI configuration registers in the MPC8555E can be programmed from the PCI bus. See Section 16.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for more information.

The PCI interface provides bus arbitration for the MPC8555E and up to five other PCI bus masters. The arbitration algorithm is a programmable two-level, round-robin priority selector. The on-chip PCI arbiter can operate in both host and agent modes or it can be disabled to allow for an external PCI arbiter.

The MPC8555E also provides an address translation mechanism to map inbound PCI to OCeaN accesses and outbound OCeaN to PCI accesses.

16.1.1.1 MPC8555E as a PCI Initiator

Upon detecting an OCeaN-to-PCI transaction, the MPC8555E requests the use of the PCI bus. For OCeaN-to-PCI bus write operations, the MPC8555E requests mastership of the PCI bus when the source completes the write operation to the OCeaN. For OCeaN-to-PCI read operations, the MPC8555E requests mastership of the PCI bus when it decodes that the access is for PCI address space.

Once granted, the MPC8555E drives the address (PCI1_AD[63:0] or PCI*n*_AD[31:0]) and the bus command (PCI1_C/$\overline{\text{BE}}$[7:0] or PCI*n*_C/$\overline{\text{BE}}$[3:0]) signals.

The master part of the interface can initiate master-abort cycles, recognizes target-abort, target-retry, and target-disconnect cycles, and supports various device selection timings. The master interface does not run fast back-to-back or exclusive accesses.

16.1.1.2 MPC8555E as a PCI Target

Upon detection of a PCI address phase, the MPC8555E decodes the address and bus command to determine if the transaction is within the local memory access boundaries. If the transaction is destined for local memory, the target interface latches the address, decodes the PCI bus command, and forwards the transaction to the OCeaN control unit. On writes to local memory, data is forwarded along with the byte enables (if applicable) to the internal control unit. On reads, the data is driven on the bus and the byte enables (if applicable) determine which byte lanes contain meaningful data.

The target interface of the MPC8555E can issue target-abort, target-retry, and target-disconnect cycles. The target interface supports fast back-to-back transactions. The target interface uses the fastest device selection timing.

The MPC8555E supports data streaming to and from local memory. This means that data can flow between the MPC8555E PCI interface and local memory as long as the internal buffers are not filled.

16.1.2 Features

The following is a list of PCI features that is supported:

- PCI interface 2.2 compatible
- 66- and 33-MHz support
- 64- and 32-bit PCI interface support
- Host and agent mode support on PCI1; host mode only on PCI2
- 64-bit dual address cycle (DAC) support
- On-chip arbitration with support for five high priority request and grant signal pairs
- Support for accesses to all PCI memory and I/O address spaces
- Support for PCI to memory and memory to PCI streaming
- Memory prefetching of PCI read accesses
- Support posting of processor to PCI and PCI to memory writes
- Support selectable snoop for Inbound accesses
- PCI configuration registers
- PCI 3.3-V compatible

16.1.3 Modes of Operation

A number of parameters that affect the PCI controller modes of operation are determined at power-on reset (POR) by reset configuration signals as described in Chapter 4, “Reset, Clocking, and Initialization.” Table 16-1 provides a summary of these modes.

Table 16-1. POR Parameters for PCI Controller

Parameter	Description	Section/Page
Host/agent configuration	Selects between host and agent mode for the PCI1 interface.	4.4.3.4/4-13
PCI Interface selection	Selects between one 64-bit PCI interface or two 32-bit PCI interfaces.	4.4.3.11/4-17

Table 16-1. POR Parameters for PCI Controller (continued)

Parameter	Description	Section/Page
PCI clocking	Selects between asynchronous or synchronous clocking for PCI	4.4.3.10/4-16
PCI arbiter enable	Enables the on-chip PCI bus arbiter	4.4.3.13/4-18
PCI output hold	Selects the number of buffer delays for PCI output signals.This provides flexibility in meeting minimum output hold specifications relative to SYSCLK for PCI	4.4.3.17/4-19
PCI I/O impedance	Selects the impedance of the PCI I/O drivers	4.4.3.12/4-17
PCI debug mode enable	Selects between normal operation or debug mode for the PCI1_AD[62:58] signals.	4.4.3.14/4-18

16.1.3.1 Host/Agent Mode Configuration

The PCI1 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). Additionally, the PCI1 controller can operate in agent configuration lock mode.The PCI2 interface only supports host mode. Note that host/agent mode selection is determined at power-up as summarized in Section 16.5.1, “Power-On Reset Configuration Modes.”

16.1.3.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). See Section 16.5.1.1, “Host Mode,” for more information.

16.1.3.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. See Section 16.5.1.2, “Agent Mode,” for more information. 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.

16.1.3.1.3 Agent Configuration Lock Mode

When the device powers up in agent configuration lock mode, it retries inbound configuration accesses until the ACL bit in the PCI bus function register is cleared. See Section 16.5.1.3, “Agent Configuration Lock Mode,” for more information.

16.1.3.2 PCI-64 or Two PCI-32 Interface Configuration

The interface can be configured to be one PCI-64 interface or two independent PCI-32 interfaces. See Section 16.5.2, “Extended 64-Bit PCI1 Signal Connections,” on page 16-76 for more information on pin assignments for the PCI-64 configuration. The initial value for PCI Interface Selection is determined by the value on the $\overline{\text{PCI2\_FRAME}}$ power-on reset configuration signal. See Chapter 4, “Reset, Clocking, and Initialization,” and the appropriate hardware specifications for more information.

16.1.3.3 PCI Clocking Configuration

The interface can be configured to be clocked asynchronously with a PCI_CLK input or synchronously with the SYSCLK input. The initial value for PCI1 clocking is determined by the value on the TSEC2_TXD1 power-on reset configuration signal. The initial value for PCI2 clocking is determined by the value on the TSEC2_TXD0 power-on reset configuration signal. See Section 4.4.3.10, "PCI Clock Selection," and the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for more information.

16.1.3.4 PCI Arbiter (Internal/External Arbiter) Configuration

The interface can be configured to use an on-chip or off-chip PCI arbiter. The initial value for PCI1 arbiter is determined by the value on the PCI1_GNT2 power-on reset configuration signal. The initial value for PCI2 Arbiter is determined by the value on the PCI2_GNT2 power-on reset configuration signal. See Section 4.4.3.13, "PCI Arbiter Configuration," and the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for more information.

16.1.3.5 PCI Signal Output Hold Configuration

The interface has a programmable output hold delay for PCI bus signals to help meet minimum output hold specifications relative to SYSCLK for PCI systems. The initial value for PCI1 Output Hold is determined by the value on the PCI1_GNT4 power-on reset configuration signal. The initial value for PCI2 Output Hold is determined by the value on the PCI2_GNT4 power-on reset configuration signal. Note that in 64-bit mode, PCI1_GNT4 controls the lower half of the AD bus (AD[31:0]) and PCI2_GNT4 controls the upper half of the AD bus (AD[63:32]). These must have the same value or the output hold will be different for the upper and lower halves of the bus. See Chapter 4, "Reset, Clocking, and Initialization," and the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for more information on these values and signal timing.

16.1.3.6 PCI Impedance Configuration

The interface has a programmable impedance for PCI bus signals. The initial value for PCI1 Impedance is determined by the value on the PCI1_GNT1 power-on reset configuration signal. The initial value for PCI2 Impedance is determined by the value on the PCI2_GNT1 power-on reset configuration signal. Note that in 64-bit mode, PCI1_GNT1 controls the lower half of the AD bus (AD[31:0]) and PCI2_GNT1 controls the upper half of the AD bus (AD[63:32]). These must have the same value or the impedance will be different for the upper and lower halves of the bus. See Chapter 4, "Reset, Clocking, and Initialization," and the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for more information.

16.1.3.7 PCI Debug Configuration

The interface offers the ability to output source information for outbound transactions in a debug mode when PCI1 is configured as a 64-bit interface. This may be useful information for a logic analyzer to debug code. When this mode is enabled, PCI1_AD[62:58] will display debug source information for the PCI1 interface. The initial value for PCI1 Debug is determined by the value on the PCI1_GNT3 power-on reset

configuration signal. Debug information for PCI2 is not available. See Chapter 4, "Reset, Clocking, and Initialization," and the appropriate hardware specifications, for more information.

16.2 External Signal Descriptions

Figure 16-2 shows the external PCI signals.

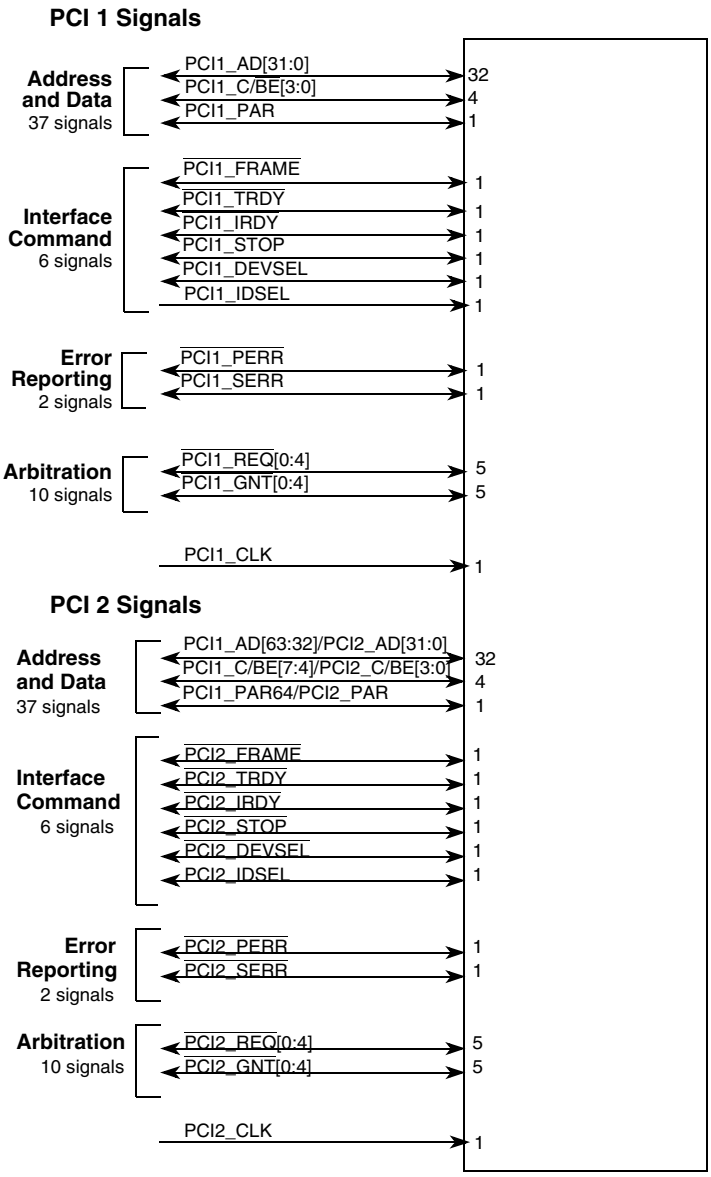


Figure 16-2. PCI Interface External Signals

Table 16-2 contains the detailed descriptions of the external PCI interface signals.

Table 16-2. PCI1 and PCI2 Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{PCI1\_ACK64}}$	I/O	64-bit transaction acknowledge. The 64-bit transaction acknowledge ($\overline{\text{PCI1\_ACK64}}$) signal is both an input and output signal. It indicates that the current target supports 64-bit transfers during the data phase of the current transaction.	
	O	As an output for the bi-directional 64-bit transaction acknowledge, this signal operates as follows:	
		State Meaning	Asserted—Indicates that this PCI controller, as the target of a PCI transaction, may use the full 64-bit data bus for the data phase of the transaction. Negated—Indicates that this PCI controller, as the target of a PCI transaction, may use only 32 bits of the data bus in servicing a data transfer.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As an input for the bi-directional 64-bit transaction acknowledge, this signal operates as follows:	
		State Meaning	Asserted—Indicates that the target of a PCI transaction may use the full 64-bit data bus for the data phase of the transaction. Negated—Indicates that the target of a PCI transaction may only use 32 bits of the data bus during the data phase of the transaction.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI1_AD[31:0], PCI1_AD[63:32]/ PCI2_AD[31:0]	I/O	PCI address/data bus. The PCI address/data bus (PCI1_AD[63:0] or PCI*n*_AD[31:0]) consists of 64 or 32 signals that are both input and output signals on this PCI controller.	
	O	As outputs for the bi-directional PCI address/data bus, these signals operate as described below.	
		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. The PCI*n*_AD[7:0] signals define the LSB and PCI*n*_AD[31:0] define the MSB for the 32-bit interface; PCI1_AD[63:56] define the MSB for the 64-bit bus mode.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional PCI address/data bus, these signals operate as described below.	
		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. The PCI*n*_AD[7:0] signals define the LSB and PCI*n*_AD[31:0] define the MSB for the 32-bit interface; PCI1_AD[63:56] define the MSB for the 64-bit bus mode.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2

Table 16-2. PCI1 and PCI2 Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI1_C/$\overline{\text{BE}}$[3:0], PCI1_C/$\overline{\text{BE}}$[7:4]/ PCI2_C/$\overline{\text{BE}}$[3:0]	I/O	Command/byte enable. The command/byte enable (PCI1_C/$\overline{\text{BE}}$[7:0] or PCI*n*_C/$\overline{\text{BE}}$[3:0]) signals are both input and output signals on this PCI controller. The command encodings for PCI bus mode are described in Section 16.4.2.2, "PCI Bus Commands."	
	O	As outputs for the bi-directional command/byte enable, these signals operate as described below.	
		State Meaning	Asserted/Negated—During the address phase, PCI1_C/$\overline{\text{BE}}$[7:0] or PCI*n*_C/$\overline{\text{BE}}$[3:0] define the bus command. During the data phase, PCI1_C/$\overline{\text{BE}}$[7:0] or PCI*n*_C/$\overline{\text{BE}}$[3:0] function as byte enables to determine which byte lanes carry meaningful data. The PCI*n*_C/$\overline{\text{BE}}$0 signal applies to the LSB.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional command/byte enable, these signals operate as described below.	
		State Meaning	Asserted/Negated—During the address phase, PCI1_C/$\overline{\text{BE}}$[7:0] or PCI*n*_C/$\overline{\text{BE}}$[3:0] indicate the command that another master is sending. During the PCI bus data phase, PCI1_C/$\overline{\text{BE}}$[7:0] or PCI*n*_C/$\overline{\text{BE}}$[3:0] indicate which byte lanes are valid.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
$\overline{\text{PCI}n\text{\_DEVSEL}}$	I/O	Device select. The device select ($\overline{\text{PCI}n\text{\_DEVSEL}}$) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional device select, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller has decoded the address and is the target of the current access. Negated—Indicates that this PCI controller has decoded the address and is not the target of the current access.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional device select, these signals operate as described below.	
		State Meaning	Asserted—Indicates that some PCI agent (other than this PCI controller) has decoded its address as the target of the current access. Negated—Indicates that no PCI agent has been selected.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
$\overline{\text{PCI}n\text{\_FRAME}}$	I/O	Frame. The frame ($\overline{\text{PCI}n\text{\_FRAME}}$) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional frame, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI master, is initiating a bus transaction. While $\overline{\text{PCI}n\text{\_FRAME}}$ is asserted, data transfers may continue. Negated—If $\overline{\text{PCI}n\text{\_IRDY}}$ is asserted, indicates that the PCI transaction is in the final data phase; if $\overline{\text{PCI}n\text{\_IRDY}}$ is negated, indicates that the PCI bus is idle.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional frame, these signals operate as described below.	
		State Meaning	Asserted—Indicates that another PCI master is initiating a bus transaction. Negated—Indicates that the transaction is in the final data phase or that the bus is idle.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2

Table 16-2. PCI1 and PCI2 Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI*n*_GNT[4:0]	O	PCI bus grant. Output signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled PCI*n*_GNT0 is an input signal. Note that PCI*n*_GNT[n] is a point-to-point signal. Every master has its own bus grant signal. Also, note that these signals are also used as reset configuration signals in the MPC8555E as described in Section 4.4.3, "Power-On Reset Configuration."	
		State Meaning	Asserted—Indicates that this PCI controller has granted control of the PCI bus to agent *n*. Negated—Indicates that this PCI controller has not granted control of the PCI bus to agent *n*.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI*n*_IDSEL	I	Initialization device select. The initialization device select (PCI*n*_IDSEL) signal is an input signal on this PCI controller. It is used as a chip select during configuration read and write transactions.	
		State Meaning	Asserted—Indicates this PCI controller is being selected as a target of a configuration read or write transactions. Negated—Indicates this PCI controller is not being selected as a target of configuration read or write transactions.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI*n*_IRDY	I/O	Initiator ready. The initiator ready (PCI*n*_IRDY) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional initiator ready, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, this PCI controller asserts PCI*n*_IRDY to indicate that valid data is present on the address/data bus. During a read, this PCI controller asserts PCI*n*_IRDY to indicate that it is prepared to accept data. Negated—Indicates that 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 PCI*n*_IRDY to insert a wait cycle when it cannot provide valid data to the target. During a read, this PCI controller negates PCI*n*_IRDY to insert a wait cycle when it cannot accept data from the target.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional initiator ready, these signals operate as described below.	
		State Meaning	Asserted—Indicates another PCI master is able to complete the current data phase of a transaction. Negated—If PCI*n*_FRAME is asserted, indicates a wait cycle from another master. If PCI*n*_FRAME is negated, indicates the PCI bus is idle.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2

Table 16-2. PCI1 and PCI2 Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI*n*_PAR	I/O	PCI parity. The PCI parity (PCI*n*_PAR) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional PCI parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity across the PCI*n*_AD[31:0] and PCI*n*_C/$\overline{\text{BE}}$[3:0] signals during address and data phases. Negated—Indicates even parity across the PCI*n*_AD[31:0] and PCI*n*_C/$\overline{\text{BE}}$[3:0] signals during address and data phases.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification,* Rev 2.2
	I	As inputs for the bi-directional PCI parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity driven by another PCI master or the PCI target during read data phases. Negated—Indicates even parity driven by another PCI master or the PCI target during read data phases.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification,* Rev 2.2
PCI1_PAR64	I/O	Upper DWORD parity. The PCI parity (PCI1_PAR64) signal is both an input and output signal on this PCI controller. PCI1_PAR64 is the even parity bit that protects the upper 32 bits of data and upper 4 bits of command/byte enable.	
	O	As outputs for the bi-directional Upper DWORD Parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity across the PCI1_AD[63:32] and PCI1_C/$\overline{\text{BE}}$[7:4] signals during address and data phases. Negated—Indicates even parity across the PCI1_AD[63:32] and PCI1_C/$\overline{\text{BE}}$[7:4] signals during address and data phases.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional Upper DWORD Parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity driven by another PCI master or the PCI target during read data phases. Negated—Indicates even parity driven by another PCI master or the PCI target during read data phases.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification*, Rev 2.2

Table 16-2. PCI1 and PCI2 Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI*n*_PERR	I/O	PCI parity error. The PCI parity error (PCI*n*_PERR) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional PCI parity error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI agent, detected a data parity error. (The PCI initiator drives PCI*n*_PERR on read operations; the PCI target drives PCI*n*_PERR on write operations.) Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional PCI parity error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that another PCI agent detected a data parity error while this PCI controller was sourcing data (this PCI controller was acting as the PCI initiator during a write, or was acting as the PCI target during a read). Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI*n*_REQ[4:0]	I	PCI bus request**.** Input signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled, PCI*n*_REQ[0] is an output. Note that PCI*n*_REQ[n] is a point-to-point signal. Every master has its own bus request signal. Following is the state meaning for the PCI*n*_REQ[n] input.	
		State Meaning	Asserted—Indicates that agent *n* is requesting control of the PCI bus to perform a transaction. Negated—Indicates that agent *n* does not require use of the PCI bus.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI1_REQ64	I/O	64-bit transaction request. Both an input and output signal on this PCI controller. It indicates that the current master desires to transfer data using 64-bit transfers. Also, note that this signal is used as a reset configuration signal in the MPC8555E as described in Section 4.4.3, “Power-On Reset Configuration."	
	O	As an output for the bi-directional 64-bit transaction request, this signal operates as follows:	
		State Meaning	Asserted—Indicates that this PCI controller, as the master of a PCI transaction, desires to use all 64 bits. Negated— Indicates that this PCI controller, as the master of a PCI transaction, uses only 32 bits of the data bus in servicing a data transfer.
		Timing	Assertion/Negation—As specified by the PCI Local Bus Specification, Rev 2.2
	I	As an input for the bi-directional 64-bit transaction request, this signal operates as described below.	
		State Meaning	Asserted—Indicates that the master of a PCI transaction is requesting to use the full 64-bit data bus for the data phase of the transaction. Negated—Indicates that the master of a PCI transaction uses only 32 bits of the data bus during the data phase of the transaction.
		Timing	Assertion/Negation—As specified by the PCI Local Bus Specification, Rev 2.2

Table 16-2. PCI1 and PCI2 Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI*n*_SERR	I/O	PCI system error.The PCI system error (PCI*n*_SERR) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional PCI system error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that 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—Indicates no error.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional PCI system error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that a target (other than this PCI controller) has detected a catastrophic error. Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI*n*_STOP	I/O	Stop.The stop (PCI*n*_STOP) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional stop, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI target, is requesting that the initiator stop the current transaction. Negated—Indicates that the current transaction can continue.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional stop, these signals operate as described below.	
		State Meaning	Asserted—Indicates that a target is requesting that the PCI initiator stop the current transaction. Negated—Indicates that the current transaction can continue.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
PCI*n*_TRDY	I/O	Target ready. Both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional target ready, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, this PCI controller asserts PCI*n*_TRDY to indicate that valid data is present on PCI*n*_AD[31:0]. During a write, this PCI controller asserts PCI*n*_TRDY to indicate that it is prepared to accept data. Negated—Indicates that 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 PCI*n*_TRDY to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates PCI*n*_TRDY to insert a wait cycle when it cannot accept data from the initiator.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2
	I	As inputs for the bi-directional target ready, these signals operate as described below.	
		State Meaning	Asserted—Another PCI target is able to complete the current data phase of a transaction. Negated—Indicates a wait cycle from another target.
		Timing	Assertion/Negation—As specified by the *PCI Local Bus Specification*, Rev 2.2

16.3 Memory Map/Register Definitions

The PCI interface unit of the MPC8555E supports the following two types of registers:

- Memory-mapped registers—These registers control PCI address translation, PCI error management, and PCI configuration register access on the MPC8555E. These registers are described in Section 16.3.1, "PCI Memory Mapped Registers," and its subsections.
- PCI configuration registers contained within the PCI configuration header—These registers are specified by the PCI bus specification for every PCI device. These registers are described in Section 16.3.2, "PCI Configuration Header," and its subsections.

16.3.1 PCI Memory Mapped Registers

The PCI memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PCSRBAR on the PCI side) plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI configuration data register, PCI CFG_DATA) must only be accessed as 32-bit quantities.

Table 16-3 lists the memory-mapped registers. Note that the memory-mapped registers for the PCI1 interface begin at offset 0x0_8000 and the memory-mapped registers for the PCI2 interface begin at offset 0x0_9000. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 16-3. PCI Memory-Mapped Register Map

Offset	Register	Access	Reset	Section/Page
PCI1 Configuration Access Registers				
0x0_8000	CFG_ADDR—PCI1 configuration address	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8004	CFG_DATA—PCI1 configuration data	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8008	INT_ACK—PCI1 interrupt acknowledge	R	0x0000_0000	16.3.1.1.3/16-19
0x0_800C–0x0_8BFC	Reserved	—	—	—
PCI1 ATMU Registers—Outbound and Inbound				
0x0_8C00–0x0_8C3C–Outbound Window 0 (default)				
0x0_8C00	POTAR0—PCI1 outbound window 0 (default) translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C04	POTEAR0—PCI1 outbound window 0 (default) translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C08	Reserved	—	—	
0x0_8C0C	Reserved	—	—	
0x0_8C10	POWAR0—PCI1 outbound window 0 (default) attributes register	R/W	0x8004_401F	16.3.1.2.4/16-22
0x0_8C14–0x0_8C1C	Reserved	—	—	

Table 16-3. PCI Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C20–0x0_8C3C—Outbound Window 1				
0x0_8C20	POTAR1—PCI1 outbound window 1 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C24	POTEAR1—PCI1 outbound window 1 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C28	POWBAR1—PCI1 outbound window 1 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C2C	Reserved	—	—	
0x0_8C30	POWAR1—PCI1 outbound window 1 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C34–0x0_8C3C	Reserved	—	—	
0x0_8C40–0x0_8C5C—Outbound Window 2				
0x0_8C40	POTAR2—PCI1 outbound window 2 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C44	POTEAR2—PCI1 outbound window 2 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C48	POWBAR2—PCI1 outbound window 2 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C4C	Reserved	—	—	
0x0_8C50	POWAR2—PCI1 outbound window 2 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C54–0x0_8C5C	Reserved	—	—	
0x0_8C60–0x0_8C7C—Outbound Window 3				
0x0_8C60	POTAR3—PCI1 outbound window 3 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C64	POTEAR3—PCI1 outbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C68	POWBAR3—PCI1 outbound window 3 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C6C	Reserved	—	—	
0x0_8C70	POWAR3—PCI1 outbound window 3 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C74–0x0_8C7C	Reserved	—	—	
0x0_8C80–0x0_8C9C—Outbound Window 4				
0x0_8C80	POTAR4—PCI1 outbound window 4 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-20
0x0_8C84	POTEAR4—PCI1 outbound window 4 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-20
0x0_8C88	POWBAR4—PCI1 outbound window 4 base address register	R/W	0x0000_0000	16.3.1.2.3/16-21
0x0_8C8C	Reserved	—	—	
0x0_8C90	POWAR4—PCI1 outbound window 4 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
0x0_8C94–0x0_8D9C	Reserved	—	—	

Table 16-3. PCI Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8DA0–0x0_8DBC–Inbound Window 3				
0x0_8DA0	PITAR3—PCI1 inbound window 3 translation address register	R/W	0x0000_0000	16.3.1.3.1/16-24
0x0_8DA4	Reserved	—	—	
0x0_8DA8	PIWBAR3—PCI1 inbound window 3 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DAC	PIWBEAR3—PCI1 inbound window 3 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DB0	PIWAR3—PCI1 inbound window 3 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DB4–0x0_8DBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC–Inbound Window 2				
0x0_8DC0	PITAR2—PCI1 inbound window 2 translation address register	R/W	0x0000_0000	16.3.1.3.1/16-24
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI1 inbound window 2 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DCC	PIWBEAR2—PCI1 inbound window 2 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DD0	PIWAR2—PCI1 inbound window 2 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DD4–0x0_8DDC	Reserved	—	—	
0x0_8DE0–0x0_8DFC–Inbound Window 1				
0x0_8DE0	PITAR1—PCI1 inbound window 1 translation address register	R/W	0x0000_0000	16.3.1.3.1/16-24
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PIWBAR1—PCI1 inbound window 1 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PIWAR1—PCI1 inbound window 1 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DF4–0x0_8DFC	Reserved	—	—	
PCI1 Error Management Registers				
0x0_8E00	ERR_DR—PCI1 error detect register	Special	0x0000_0000	16.3.1.4.1/16-29
0x0_8E04	ERR_CAP_DR—PCI1 error capture disabled register	R/W	0x0000_0000	16.3.1.4.2/16-30
0x0_8E08	ERR_EN—PCI1 error enable register	R/W	0x0000_0000	16.3.1.4.3/16-31
0x0_8E0C	ERR_ATTRIB—PCI1 error attributes capture register	R/W	0x0000_0000	16.3.1.4.4/16-32
0x0_8E10	ERR_ADDR—PCI1 error address capture register	R/W	0x0000_0000	16.3.1.4.5/16-33
0x0_8E14	ERR_EXT_ADDR—PCI1 error extended address capture register	R/W	0x0000_0000	16.3.1.4.6/16-33
0x0_8E18	ERR_DL—PCI1 error data low capture register	R/W	0x0000_0000	16.3.1.4.7/16-33

Table 16-3. PCI Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8E1C	ERR_DH—PCI1 error data high capture register	R/W	0x0000_0000	16.3.1.4.8/16-34
0x0_8E20	GAS_TIMR—PCI1 gasket timer register	R/W	0x0000_0000	16.3.1.4.9/16-35
0x0_8E24–0x0_8EFC	Reserved	—	—	
0x0_8F00–0x0_8FFC	Reserved for debug	—	—	
0x0_9000–0x0_9FFC	PCI2 registers **Note:** The PCI2 Interface has the same memory-mapped registers that are described for PCI1 from 0x0_8000 to 0x0_8FFF except the offsets are from 0x0_9000 to 0x0_9FFF.			

16.3.1.1 PCI Configuration Access Registers

The PCI configuration header, shown in Figure 16-24 and Figure 16-58, is accessed through an indirect method utilizing a pair of 32-bit memory-mapped access registers. For PCI1, CFG_ADDR is at offset 0x0_8000 and CFG_DATA is at offset 0x0_8004. For PCI2, CFG_ADDR is at offset 0x0_9000 and CFG_DATA is at offset 0x0_9004.

16.3.1.1.1 PCI Configuration Address Register (CFG_ADDR)

The CFG_ADDR register is shown in Figure 16-3.

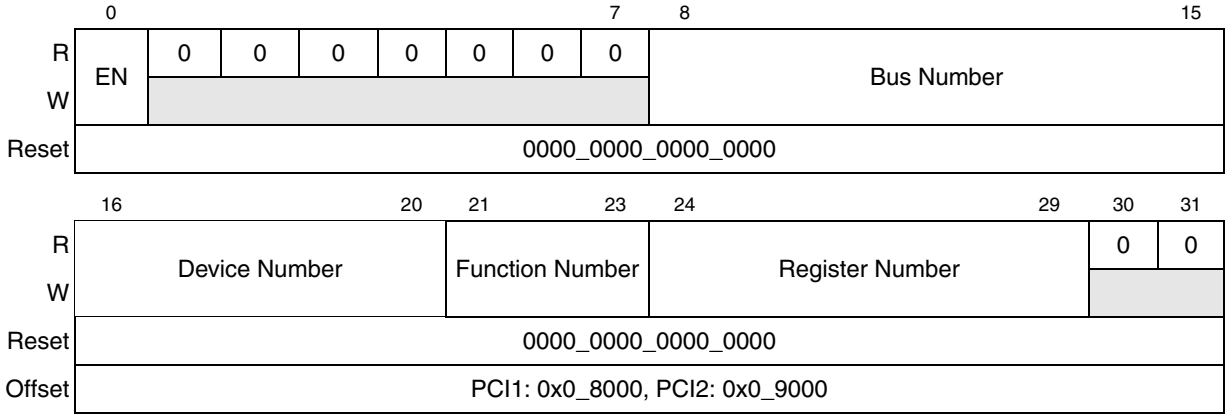


Figure 16-3. PCI CFG_ADDR Register

Table 16-4 describes the bit settings for the CFG_ADDR register.

Table 16-4. PCI CFG_ADDR Field Descriptions

Bits	Name	Description
0	EN	Enable. Allow a PCI configuration access when PCI CFG_DATA is accessed
1–7	—	Reserved
8–15	Bus Number	PCI bus number to access

Table 16-4. PCI CFG_ADDR Field Descriptions (continued)

Bits	Name	Description
16–20	Device Number	Device number to access on specified bus
21–23	Function Number	Function to access within specified device
24–29	Register Number	32-bit register to access within specified device
30–31	—	Reserved, hardwired to logic 00

Bus number 0xb00 and device number 0b0_0000 are used to configure the internal PCI configuration header of the MPC8555E itself.

See Section 16.4.2.11.2, "Accessing the PCI Configuration Space in Host Mode," and Section 16.4.2.11.3, "PCI Configuration in Agent and Agent Lock Modes," for usage of PCI CFG_ADDR.

16.3.1.1.2 PCI Configuration Data Register (CFG_DATA)

The CFG_DATA register is shown in Figure 16-3.

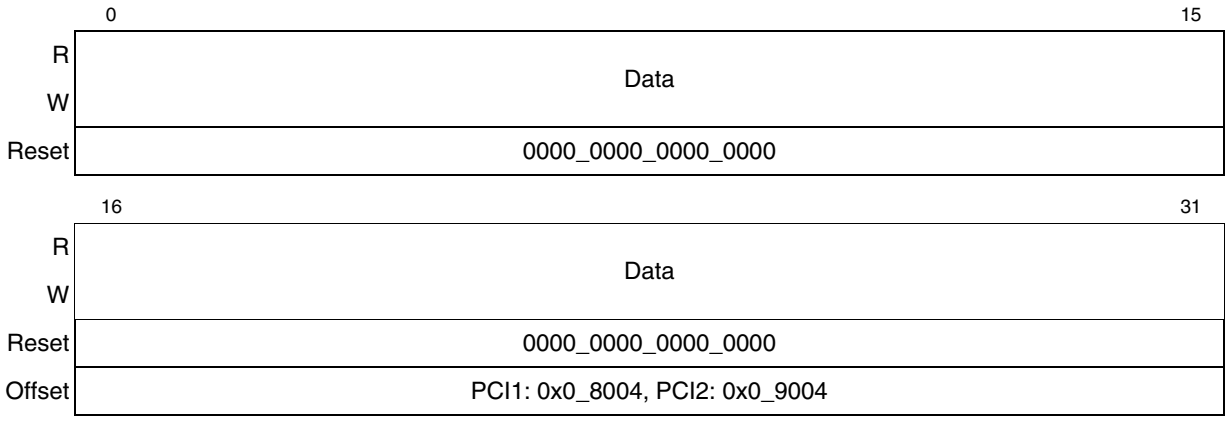


Figure 16-4. PCI CFG_DATA Register

Table 16-5 describes the bit settings for the CFG_DATA register.

Table 16-5. PCI CFG_DATA Field Descriptions

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI configuration cycle if the PCI CFG_ADDR enable bit is set. If the enable bit is not set, a PCI I/O transaction is generated.

The CFG_DATA register is a 4-byte window into the little-endian PCI configuration header data structure; therefore byte addressing within the CFG_DATA register uses little-endian convention. Note that CFG_DATA may contain 1, 2, 3, or 4 bytes depending on the size of the register being accessed.

See Section 16.4.2.11.2, "Accessing the PCI Configuration Space in Host Mode," and Section 16.4.2.11.3, "PCI Configuration in Agent and Agent Lock Modes," for usage of CFG_DATA.

16.3.1.1.3 PCI Interrupt Acknowledge Register (INT_ACK)

An external PCI interrupt acknowledge transaction is generated by reading the INT_ACK register. For PCI1, INT_ACK is at offset 0x0_8008. For PCI2, INT_ACK is at offset 0x0_9000. PCI INT_ACK is read-only and a write to that address results in nothing. The INT_ACK register is shown in Figure 16-5.

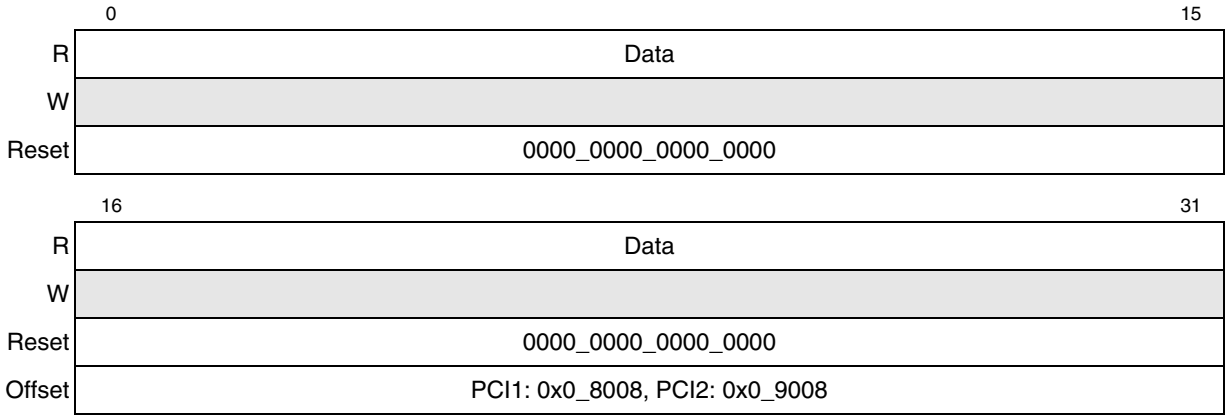


Figure 16-5. PCI INT_ACK Register

Table 16-6 describes the bit settings for the INT_ACK register.

Table 16-6. PCI INT_ACK Field Descriptions

Bits	Name	Description
0–31	Data	A read to this register generates a PCI interrupt acknowledge cycle.

16.3.1.2 PCI ATMU Outbound Registers

The outbound address translation and mapping unit controls the mapping of transactions from the internal 32-bit address space of the MPC8555E to the external PCI address space. The outbound ATMU consists of four translation windows plus a default translation for transactions that do not hit in one of the four windows.

Each window contains a base address that points to the beginning of the window in the local address map, a translation address that specifies the high-order bits of the transaction in the external PCI address space, and a set of attributes including window size and external transaction type.

Each window must be aligned based on the granularity specified by the window size. If two outbound ATMU windows overlap in the local address space, the mapping of the lower numbered window has precedence over the higher numbered window. Note that outbound translation windows must not overlap the configuration access registers.

Window 0 is the default window and is the only window enabled upon reset. The default outbound register set is used when a transaction misses in all of the other outbound windows.

16.3.1.2.1 PCI Outbound Translation Address Registers (POTAR*n*)

The PCI outbound translation address registers (POTAR*n*) select the starting addresses in the PCI address space for hits in the PCI outbound windows. The translated address is created by concatenating the transaction offset to this translation address. The format of the POTAR*n* is shown in Figure 16-6.

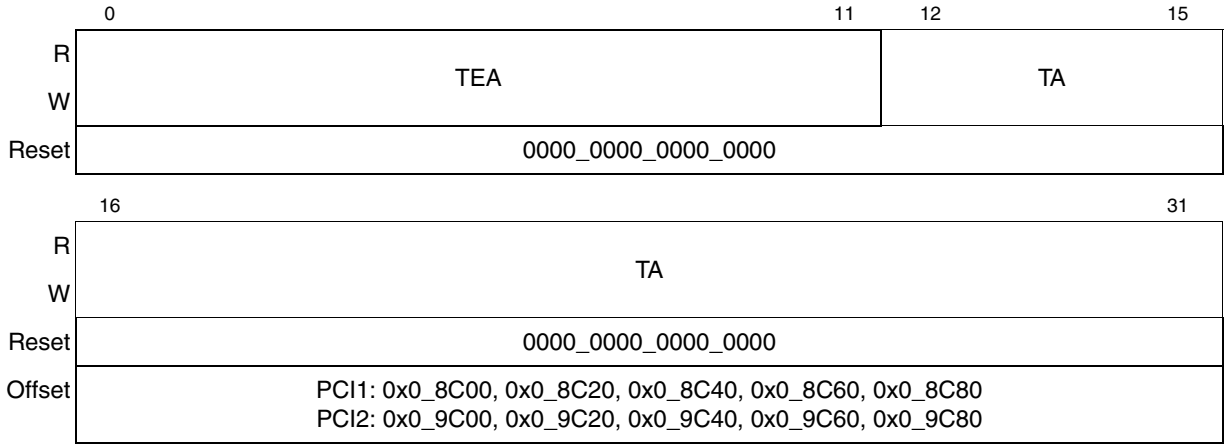


Figure 16-6. PCI Outbound Translation Address Registers

Table 16-7 describes the fields of the POTAR*n* registers.

Table 16-7. POTAR*n* Field Descriptions

Bits	Name	Description
0–11	TEA	Translation extended address. Represents bits 43–32 of a 64-bit PCI address (bit 0 is lsb)
12–31	TA	Translation address. Represents bits 31–12 of the PCI address. Based on the size of the window specified in POWAR*n*[OWS], the low-order bits of this field may be ignored.

16.3.1.2.2 PCI Outbound Translation Extended Address Registers (POTEAR*n*)

The PCI outbound translation extended address registers (POTEAR*n*) contain the most significant bits of a 64-bit translation address. The format of the POTEAR*n* is shown in Figure 16-7.

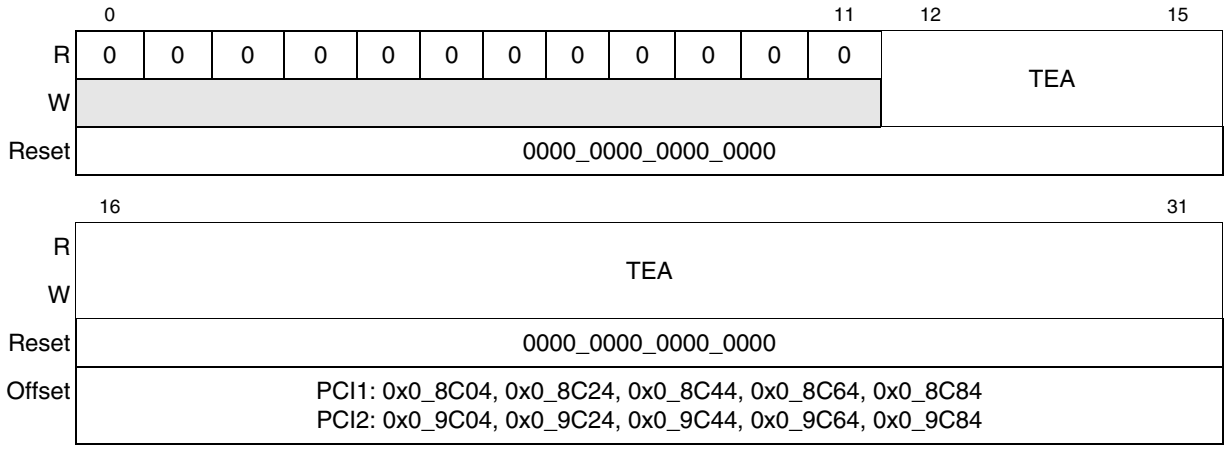


Figure 16-7. PCI Outbound Translation Extended Address Registers

Table 16-8 describes the fields of the POTEAR*n*.

Table 16-8. POTEAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. Comprises bits [44–63] of the translation address register

16.3.1.2.3 PCI Outbound Window Base Address Registers (POWBAR*n*)

The PCI outbound window base address registers (POWBAR*n*) point to the beginning of each translation window in the local 32-bit address space. Addresses for outbound transactions are compared to the appropriate bits in these registers, according to the sizes of the windows. If a transaction does not fall within one of these windows, the default translation and mapping is used. The default window is always enabled and used when the other windows miss.

Note that POWBAR0 (for outbound ATMUþwindow 0) is not used, because window 0 is the default window used when no other windows match. POWBAR0 may be read from and written to, but the value is ignored.

The format of the POWBAR*n* is shown in Figure 16-8.

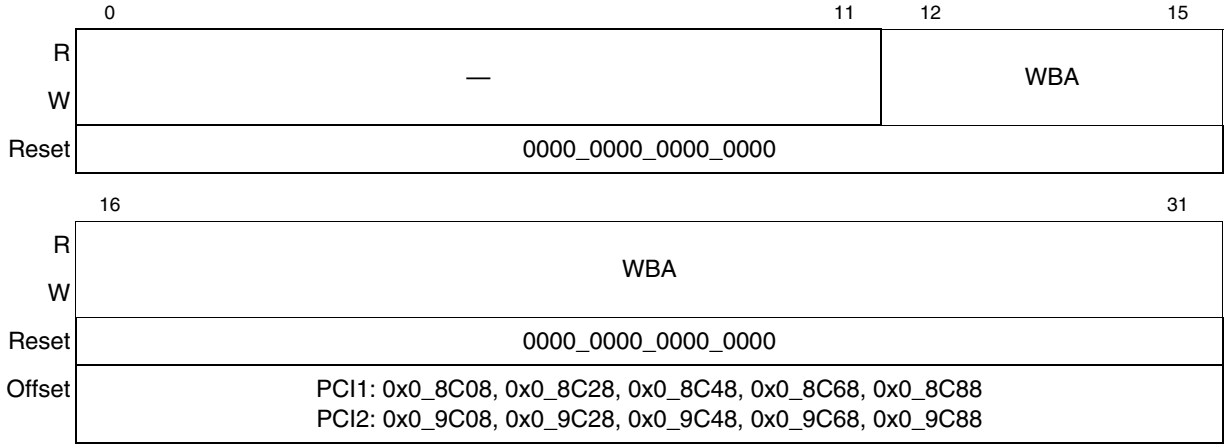


Figure 16-8. PCI Outbound Window Base Address Registers

Table 16-9 describes the field of the POWBAR*n*.

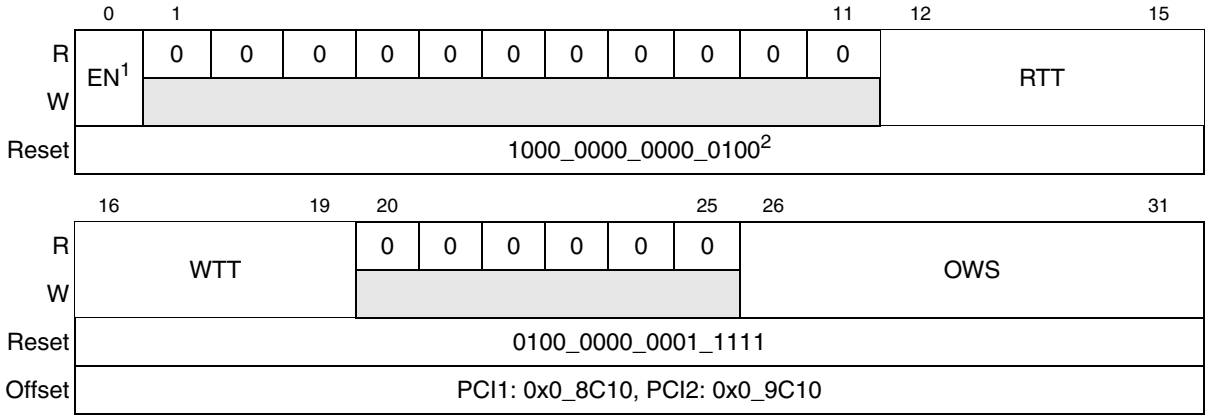
Table 16-9. POWBAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved, should be cleared.
12–31	WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits.

16.3.1.2.4 PCI Outbound Window Attributes Registers (POWAR*n*)

The PCI outbound window attributes registers (POWAR*n*) define the window sizes to translate and other attributes for the translations. The minimum window size is 4 Kbytes. The maximum window size is 4 Gbytes.

The default window attribute register, POWAR0, is shown in Figure 16-9. Note that the fields for all of the POWAR*n* registers are the same, only the reset values are different.



1 For POWAR0, translation is always enabled. The enable field (EN) may be read and written, but the value is ignored.

2 The default window is enabled, configured for memory read and memory write, and set to an OWS size of 4 Gbytes.

Figure 16-9. PCI Outbound Window Attributes Register 0 (Default)

POWAR1–POWAR4 are shown in Figure 16-10.

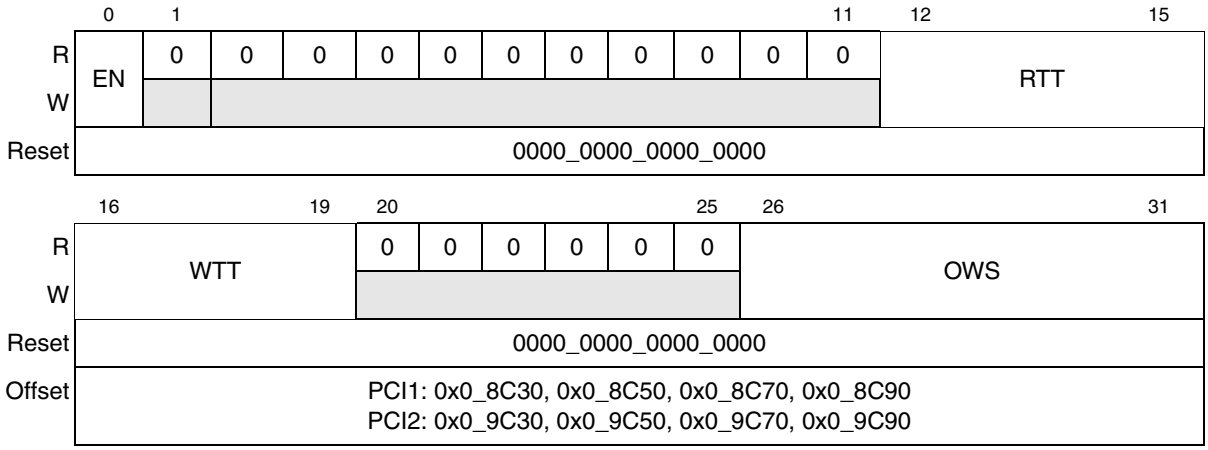


Figure 16-10. PCI Outbound Window Attributes Registers 1–4

Table 16-10 describes the fields for the POWAR*n* registers.

Table 16-10. POWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables this address translation. Note that for POWAR0,translation is always enabled. The enable field (EN) may be read and written, but the value is ignored.
1–11	—	Reserved
12–15	RTT	Read transaction type to run on PCI 0000 Reserved ... 0011 Reserved 0100 Memory read 0101 Reserved ... 0111 Reserved 1000 I/O Read 1001 Reserved ... 1111 Reserved
16–19	WTT	Write transaction type to run on PCI 0000 Reserved ... 0011 Reserved 0100 Memory write 0101 Reserved ... 0111 Reserved 1000 I/O Write 1001 Reserved ... 1111 Reserved
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4 Kbytes. 000000: Reserved ... 001011: 4-Kbyte window size 001100: 8-Kbyte window size ... 011111: 4-Gbyte window size 100000:Reserved ... 111111: Reserved The default POWAR register (0x0_8C10) has an OWS value of 011111. Also note that for POWAR0, setting OWS to less than 4 Gbytes causes addresses that miss in the other outbound windows to be aliased to the smaller address range defined by POWAR0[OWS] and POTAR0.

16.3.1.3 PCI ATMU Inbound Registers

The inbound address translation and mapping unit controls the mapping of transactions from the external PCI address space to the local address space of the MPC8555E. The inbound ATMU is comprised of four windows—a configuration window and three general translation windows. The configuration window has

higher priority than all other inbound ATMU windows and takes precedence over them if there is an overlap.

NOTE

PCI1 and PCI2 can also be used as possible target interfaces for PCI inbound ATMUs, mapping from one PCI to the other. However, it is illegal for PCI1 to use PCI1 as a target, or for PCI2 to use PCI2 for a target.

Each window contains the following:

- A base address, which points to the beginning of the window in the external PCI address map. The base address of each window is also accessible by PCI configuration transactions as base address registers within the PCI configuration header, as shown in Figure 16-24. The registers may be read or updated equivalently through the ATMU memory map or through PCI configuration transactions to the PCI configuration header.
- A translation address, which specifies the upper order bits of the transaction in the local address space
- A set of attributes including window size and internal transaction attributes

Each window's base address and translation address must be aligned to the size of the window. If two general inbound ATMU windows overlap in the external PCI address space, the mappings of the lower numbered window are applied; however, it is illegal for an inbound window to overlap the PCSRBAR window. In addition, if inbound ATMU windows are overlapped, the ATMU windows must not map to the same address with different sets of attributes.

Note that PCSRBAR in the PCI configuration header acts as a fourth inbound window that translates a 1-Mbyte region of PCI space to the local configuration space pointed to by CCSRBAR. PCSRBAR can be accessed by PCI configuration cycles or by accessing the PCI configuration header through the PCI CFG_ADDR and PCI CFG_DATA registers. See Section 16.3.1.1.1, "PCI Configuration Address Register (CFG_ADDR)," Section 16.3.1.1.2, "PCI Configuration Data Register (CFG_DATA)," and Section 16.3.2.11, "PCI Base Address Registers." All accesses to PCSRBAR have an automatic internal byte lane redirection from the little-endian PCI bus to the big-endian CCSRBAR configuration space.

16.3.1.3.1 PCI Inbound Translation Address Registers (PITAR*n*)

The PCI inbound translation address registers (PITAR*n*) points to the beginning of the local address space for the inbound window. The translated address is created by concatenating the transaction offset to this translation address. The format of the PITAR*n* is shown in Figure 16-11.

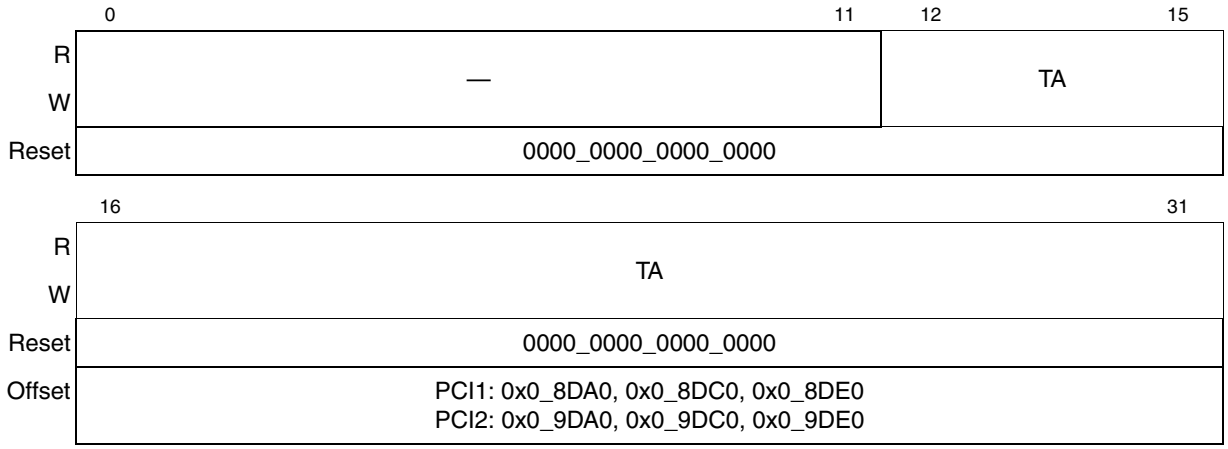


Figure 16-11. PCI Inbound Translation Address Registers

Table 16-11 describes the fields of the PITAR*n* registers.

Table 16-11. PITAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved, should be cleared.
12–31	TA	Translation address. Indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. TA corresponds to the high-order 20 bits of a 32-bit local address.

16.3.1.3.2 PCI Inbound Window Base Address Registers (PIWBAR*n*)

The PCI inbound window base address registers (PIWBAR*n*) select the PCI base address for the windows that are translated to the local address space of MPC8555E. Addresses for inbound transactions are compared to these windows. If a PCI transaction does not fall within one of these spaces, then the PCI interface does not assert DEVSEL. The PIWBAR*n* is shown in Figure 16-12.

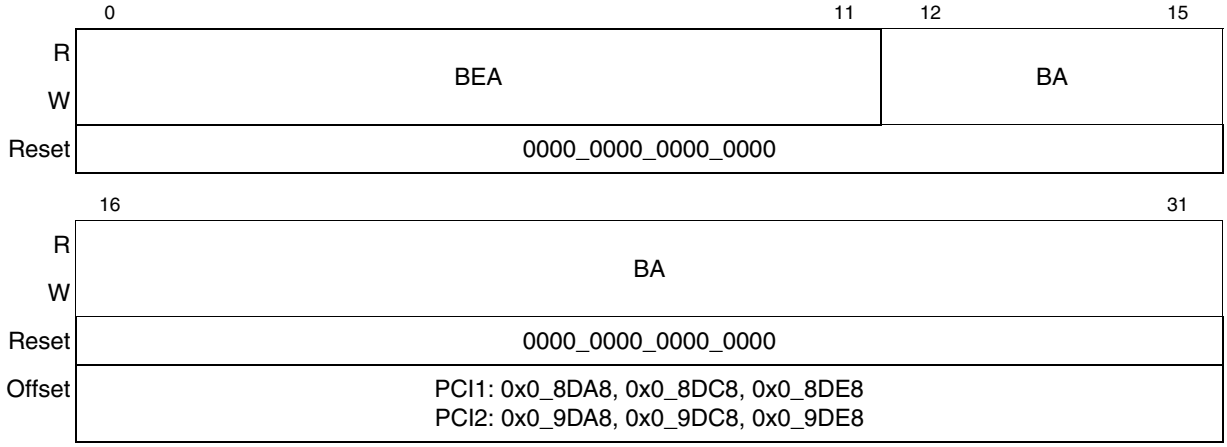


Figure 16-12. PCI Inbound Window Base Address Registers

Table 16-12 describes the fields of the PIWBAR*n* registers.

Table 16-12. PIWBAR Field Descriptions

Bits	Name	Description
0–11	BEA	Base extended address. Corresponds to bits 43–32 of a 64-bit PCI base address
12–31	BA	Base address. Corresponds to bits 31–12 of a PCI base address

16.3.1.3.3 PCI Inbound Window Base Extended Address Registers (PIWBEAR*n*)

The PCI inbound window base extended address registers (PIWBEAR*n*) contain the most significant bits of a 64-bit base address. Note that inbound window 1 supports only a 32-bit base address and does not define an inbound window base extended address register. The PIWBEAR*n* are shown in Figure 16-13.

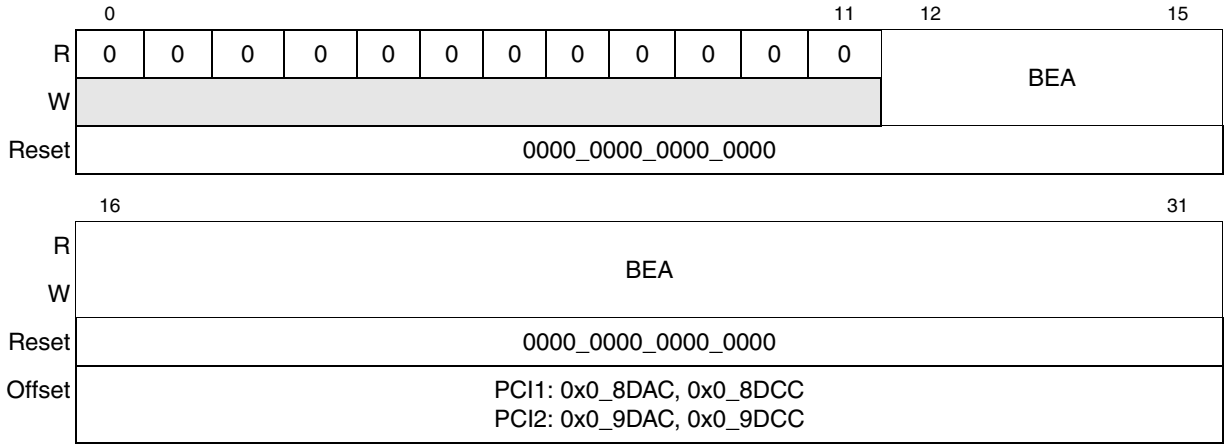


Figure 16-13. PCI Inbound Window Base Extended Address Registers

Table 16-13 describes the fields of the PIWBEAR*n* registers.

Table 16-13. PIWBEAR Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	BEA	Base extended address. Corresponds to bits 63–44 of a 64-bit PCI base address

16.3.1.3.4 PCI Inbound Window Attributes Registers (PIWAR*n*)

The PCI inbound window attributes registers (PIWAR*n*) define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The format of the PIWBAR*n* is shown in Figure 16-14.

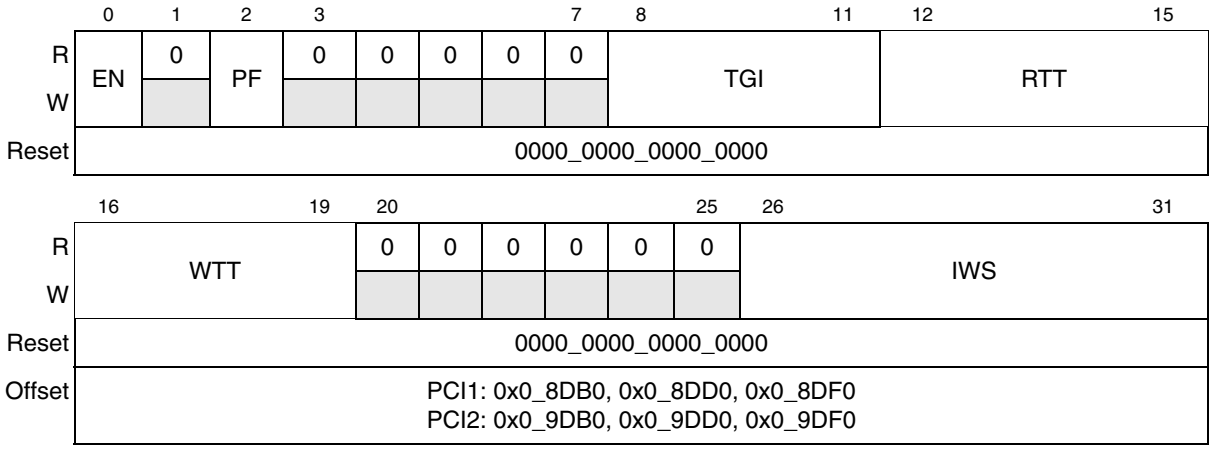


Figure 16-14. PCI Inbound Window Attributes Registers

Table 16-14 describes the fields of the PIWAR*n* registers.

Table 16-14. PIWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables this address translation
1	—	Reserved
2	PF	Prefetchable. Indicates that the address space is prefetchable so that prefetching and streaming are attempted. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved
8–11	TGI	Target interface 0000 PCI1 0001 PCI2 ... 1011 Reserved 1100–1110 Reserved 1111 Local memory (DDR SDRAM, local bus, SRAM) Note: If this field is set to an I/O port rather than local memory space, attributes for the external I/O transaction are assigned in an outbound ATMU of that I/O controller.
12–15	RTT	Read transaction type. Transaction type to run if access is a read. The field description differs subject to the transaction being targeted to I/O interface or to local memory. Following are the transaction type settings for reads to an I/O interface: 0000–0011 Reserved 0100 Read 0101–1111 Reserved Following are the transaction type settings for reads to local memory: 0000–0011 Reserved 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, unlock L2 cache line 1000–1111 Reserved

Table 16-14. PIWAR*n* Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Transaction type to run if access is a write. The field description differs subject to the transaction being targeted to an I/O interface or to local memory. Following are the transaction type settings for writes to an I/O interface: 0000–0011 Reserved 0100 Write 0101–1111 Reserved Following are the transaction type settings for writes to local memory: 0000–0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Write, allocate L2 cache line 0111 Write, allocate and lock L2 cache line 1000–1111 Reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window is 4 Kbytes. 000000–001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000–111111 Reserved For configuration and run-time registers, the window size is fixed at 010011 1-Mbyte window size For register set 0, the window size is limited to 4 Gbytes or smaller.

16.3.1.4 PCI Error Management Registers

When a PCI error is detected, the appropriate error bit is set in the PCI error detect register. Subsequent errors set the appropriate error bits in the error detection registers, but relevant information (attributes, address, and data) is captured only for the first error. The PCI error detect register is a write-1-to-clear type register. That is, reading from this register occurs normally; however, write operations are different in that the bits can be cleared but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 25 and not affect any other bits in the register, the value 0x0000_0040 is written to the register.

The error bit is set regardless of the state of the corresponding error enable bit in the PCI error enable register. The error enable bits are used to send or block the error reporting to the interrupt mechanism. The interrupt can be cleared by writing 0xFFFF_FFFF to the PCI error detect register.

Note that some errors are reported in two bits—one in the PCI error detect register (ERR_DR) and another in the PCI bus status register in the PCI configuration header. These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur. Refer to Table 16-52 for PCI mode error actions. Likewise, some errors are enabled by programming two bits—one in the PCI error enable register and another in the PCI bus command register in the PCI configuration header.

A master-abort condition during a configuration cycle is not necessarily an error. In this case, if relevant, the master abort error enable can be disabled to prevent the reporting of master-aborts during outbound configuration cycles. Master-aborts during configuration reads return 0xFFFF_FFFF.

If a data parity error occurs during an inbound configuration write access, the error is reported and captured. However, the erroneous data is written to the register specified in the transaction. Therefore, PCI data parity error recovery routines must include reinitialization of the PCI configuration register if the error occurred during a configuration write.

For an inbound configuration write transaction with a parity error, the MPC8555E always updates the register access and generates the error interrupt if the interrupt enabled bit is set.

See Section 16.4.2.13, "PCI Error Functions," for more detail on error handling.

16.3.1.4.1 PCI Error Detect Register (ERR_DR)

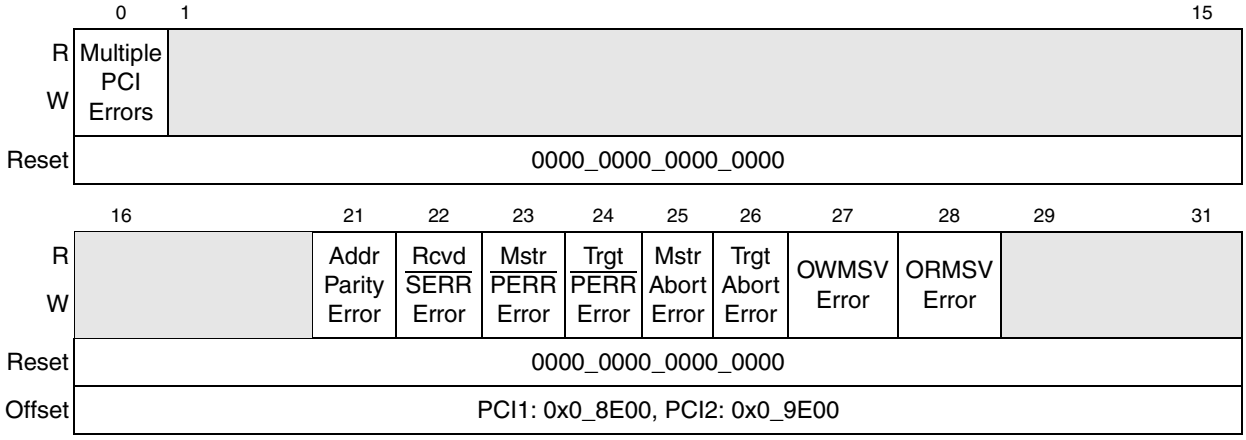


Figure 16-15. PCI Error Detect Register (ERR_DR)

Table 16-15 describes ERR_DR fields. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and an error occurs, the appropriate parity detect and master-abort bits in ERR_DR must be cleared and the appropriate enable bits in ERR_EN must be set to ensure that an interrupt is generated. See Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."

Table 16-15. ERR_DR Field Descriptions

Bits	Name	Description
0	Multiple PCI errors	0 Multiple PCI errors of the same type were not detected (write-1-to-clear) 1 Multiple PCI errors of the same type were detected
1–20	—	Reserved
21	Addr parity error	Address parity error (write-1-to-clear)
22	Rcvd $\overline{\text{SERR}}$ error	Received $\overline{\text{SERR}}$ error (write-1-to-clear)
23	Mstr $\overline{\text{PERR}}$ error	Master $\overline{\text{PERR}}$ error (write-1-to-clear)
24	Trgt $\overline{\text{PERR}}$ error	Target $\overline{\text{PERR}}$ error (write-1-to-clear)

Table 16-15. ERR_DR Field Descriptions (continued)

Bits	Name	Description
25	Mstr abort error	Master abort error (write-1-to-clear)
26	Trgt abort error	Target abort error (write-1-to-clear)
27	OWMSV error	Outbound write memory space violation error (write-1-to-clear)
28	ORMSV error	Outbound read memory space violation error (write-1-to-clear)
29–31	—	Reserved

16.3.1.4.2 PCI Error Capture Disable Register (ERR_CAP_DR)

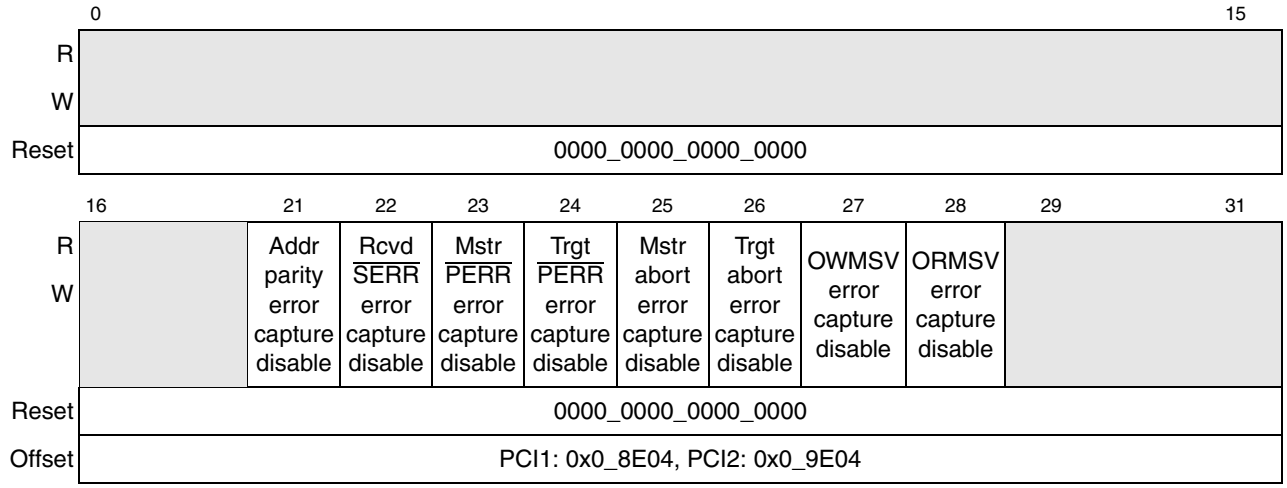


Figure 16-16. PCI Error Capture Disable Register (ERR_CAP_DR)

Table 16-16. ERR_CAP_DR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	Addr parity error capture disable	Disable capture for address parity errors
22	Rcvd $\overline{\text{SERR}}$ error capture disable	Disable capture for received $\overline{\text{SERR}}$ errors
23	Mstr $\overline{\text{PERR}}$ error capture disable	Disable capture for master $\overline{\text{PERR}}$ errors
24	Trgt $\overline{\text{PERR}}$ error capture disable	Disable capture for target $\overline{\text{PERR}}$ errors
25	Mstr abort error capture disable	Disable capture for master abort errors
26	Trgt abort error capture disable	Disable capture for target abort errors
27	OWMSV error capture disable	Disable capture for outbound write memory space violation errors
28	ORMSV error capture disable	Disable capture for outbound read memory space violation errors
29–31	—	Reserved

16.3.1.4.3 PCI Error Enable Register (ERR_EN)

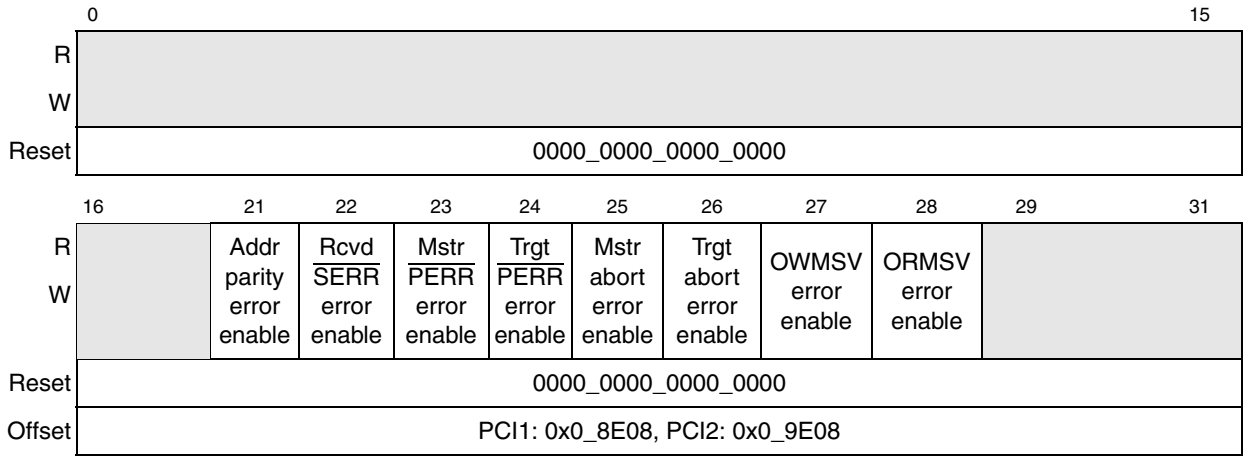


Figure 16-17. PCI Error Enable Register (ERR_EN)

Table 16-17 describes ERR_DR fields. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, the appropriate parity detect and master-abort bits in ERR_DR must be cleared and the appropriate enable bits in ERR_EN must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."

Table 16-17. ERR_EN Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	Addr parity error enable	Enable reporting address parity errors
22	Rcvd $\overline{\text{SERR}}$ error enable	Enable reporting received $\overline{\text{SERR}}$ errors
23	Mstr $\overline{\text{PERR}}$ error enable	Enable reporting master $\overline{\text{PERR}}$ errors
24	Trgt $\overline{\text{PERR}}$ error enable	Enable reporting target $\overline{\text{PERR}}$ errors
25	Mstr abort error enable	Enable reporting master abort errors
26	Trgt abort error enable	Enable reporting target abort errors
27	OWMSV error enable	Enable reporting outbound write memory space violation errors
28	ORMSV error enable	Enable reporting outbound read memory space violation errors
29–31	—	Reserved

16.3.1.4.4 PCI Error Attributes Capture Register (ERR_ATTRIB)

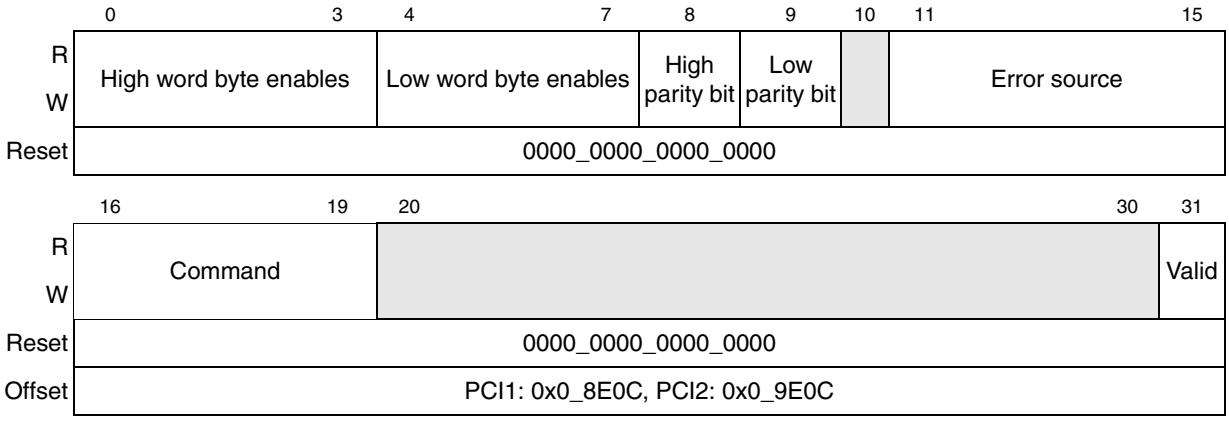


Figure 16-18. PCI Error Attributes Capture Register (ERR_ATTRIB)

Table 16-18. ERR_ATTRIB Field Descriptions

Bits	Name	Description
0–3	High word byte enables	PCI byte enables for most significant word of the double word
4–7	Low word byte enables	PCI byte enables for least significant word of the double word
8	High parity bit	Parity bit for most significant PCI bus data word (only valid for 64-bit PCI bus)
9	Low parity bit	Parity bit for least significant PCI bus data word
10	—	Reserved
11–15	Error source	The source of the PCI transaction 00000 PCI interface 1 00001 PCI interface 2 00010 Reserved 00011 Reserved 00100 Local bus controller 00101 Reserved 00110 Reserved 00111 Security 01000 Reserved 01001 Reserved 01010 Reserved 01011 Reserved 01100 Reserved 01101 Reserved 01110 Reserved 01111 Reserved 10000 e500 core (instruction) 10001 e500 core (data) 10010 Reserved 10011 Reserved 10100 CPM 10101 DMA 10110 RDC 10111 SAP 11000 TSEC1 11001 TSEC2 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved
16–19	Command	PCI command
20–30	—	Reserved
31	Valid info	The PCI bus capture registers contain valid information

16.3.1.4.5 PCI Error Address Capture Register (ERR_ADDR)

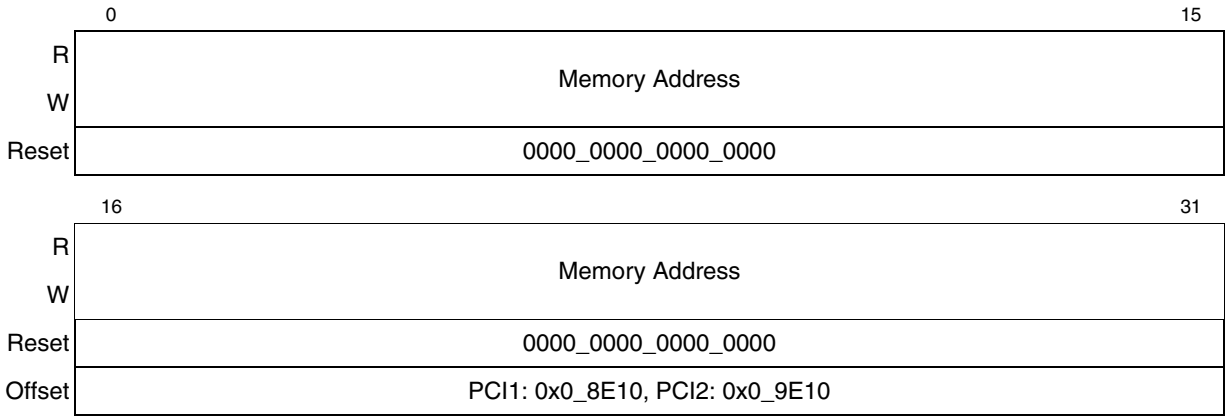


Figure 16-19. PCI Error Address Capture Register (ERR_ADDR)

Table 16-19. ERR_ADDR Field Descriptions

Bits	Name	Description
0–31	Memory address	Memory transaction address

16.3.1.4.6 PCI Error Extended Address Capture Register (ERR_EXT_ADDR)

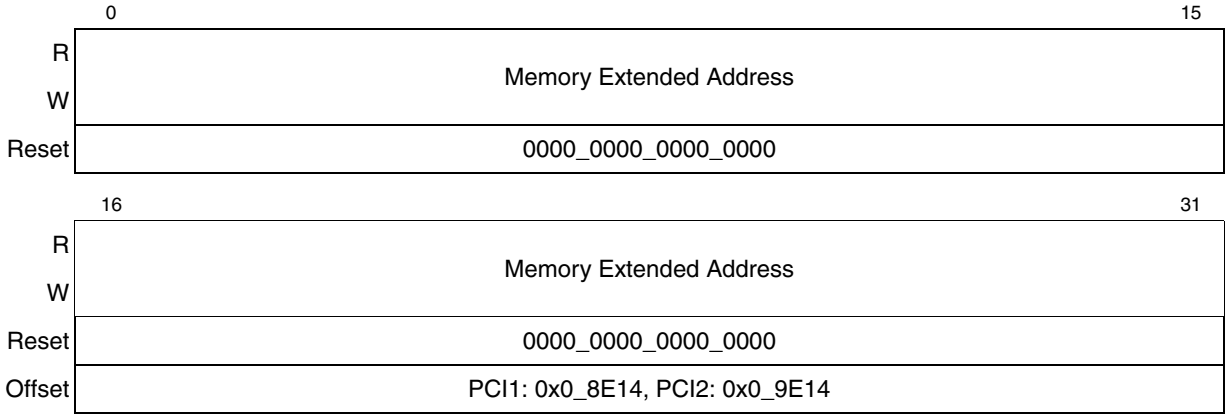


Figure 16-20. PCI Error Extended Address Capture Register (ERR_EXT_ADDR)

Table 16-20. ERR_EXT_ADDR Field Descriptions

Bits	Name	Description
0–31	Memory extended address	Memory transaction extended address

16.3.1.4.7 PCI Error Data Low Capture Register (ERR_DL)

Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.

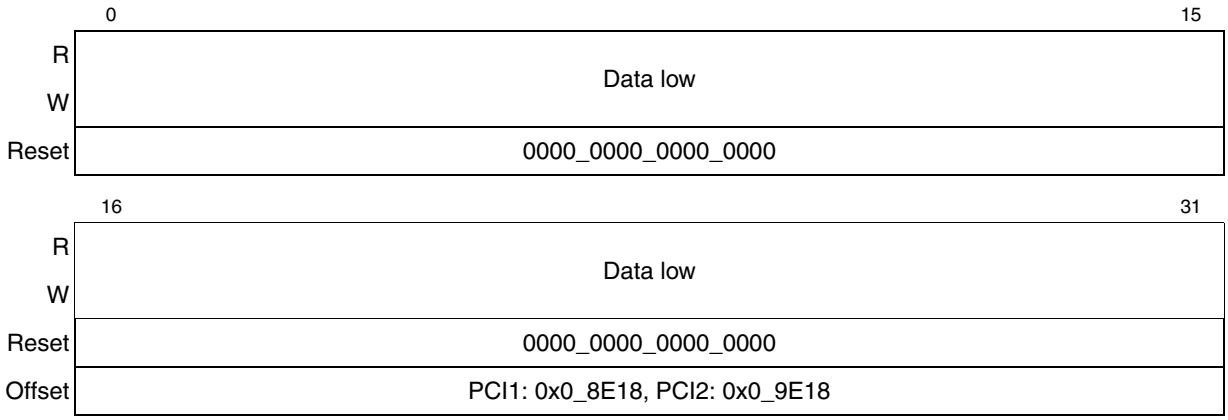


Figure 16-21. PCI Error Data Low Capture Register (ERR_DL)

Table 16-21. ERR_DL Field Descriptions

Bits	Name	Description
0–31	Data low	Least significant PCI bus data

16.3.1.4.8 PCI Error Data High Capture Register (ERR_DH)

Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.

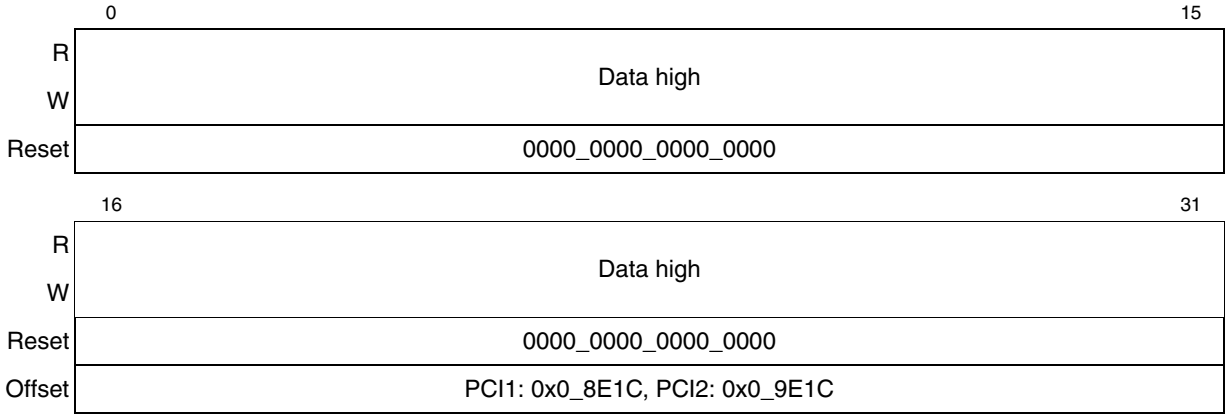


Figure 16-22. PCI Error Data High Capture Register (ERR_DH)

Table 16-22. ERR_DH Field Descriptions

Bits	Name	Description
0–31	Data high	Most significant PCI bus data word (only valid with 64-bit PCI bus)

16.3.1.4.9 PCI Gasket Timer Register (GAS_TIMR)

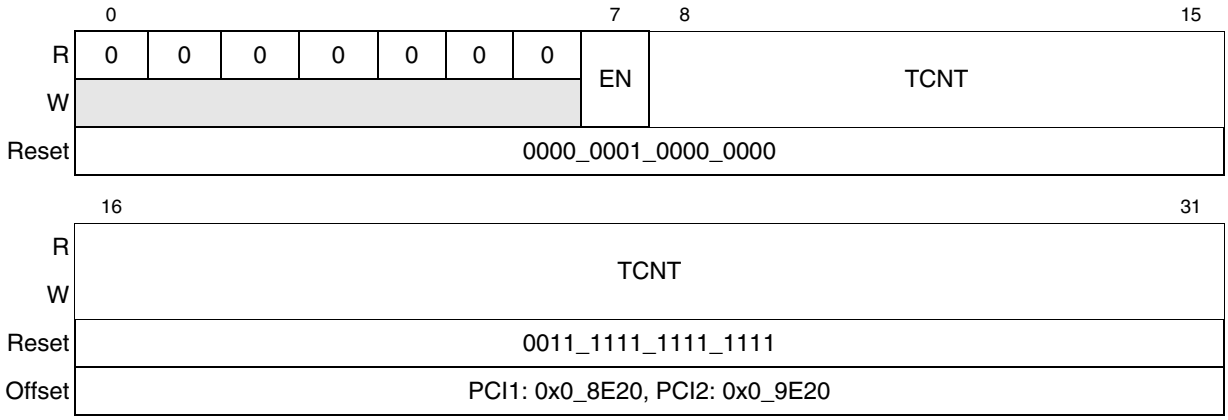


Figure 16-23. PCI Gasket Timer Register (GAS_TIMR)

Table 16-23. GAS_TIMR Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	EN	Gasket timer enable PCI default: Gasket timer is enabled
8–24	TCNT	Number of system clocks to purge a non-prefetchable inbound read buffer

16.3.2 PCI Configuration Header

The *PCI Local Bus Specification* defines the configuration registers contained within the PCI configuration header from 0x00 through 0x3F.

Figure 16-24 lists the common PCI configuration header as implemented by the MPC8555E.

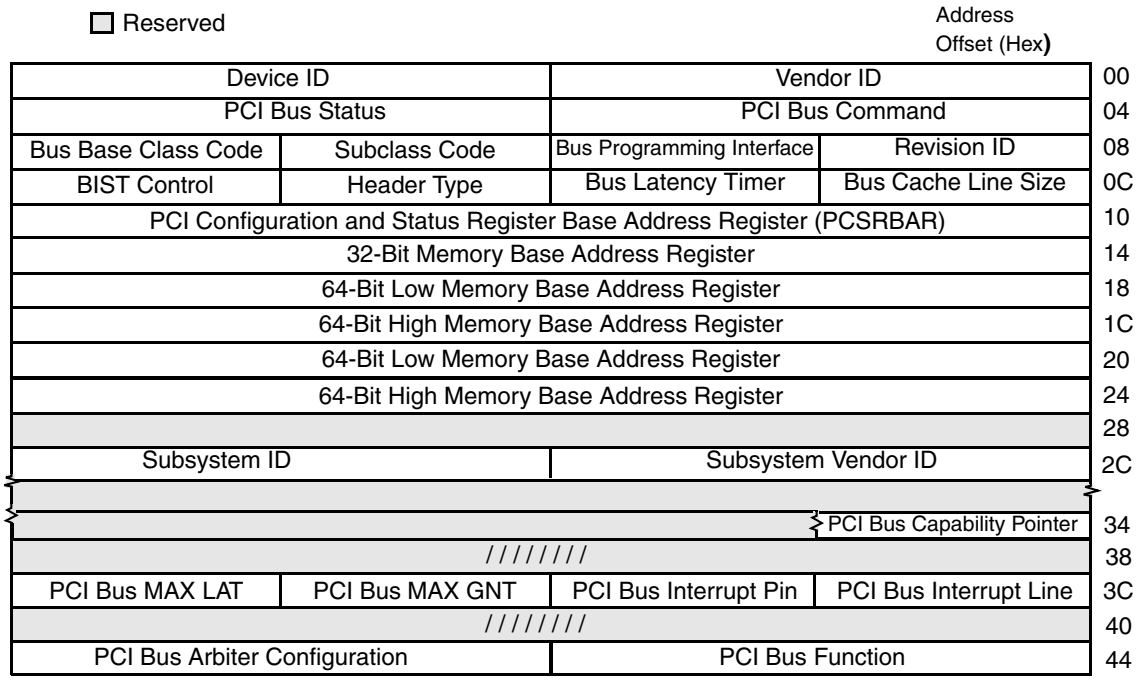


Figure 16-24. MPC8555E PCI Configuration Header

Table 16-49 in Section 16.4.2.11.1, "PCI Configuration Space Header," provides a summary of the PCI configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*.

16.3.2.1 PCI Vendor ID Register—Offset 0x00

The PCI vendor ID register, shown in Figure 16-25, is used to identify the manufacturer of the part.

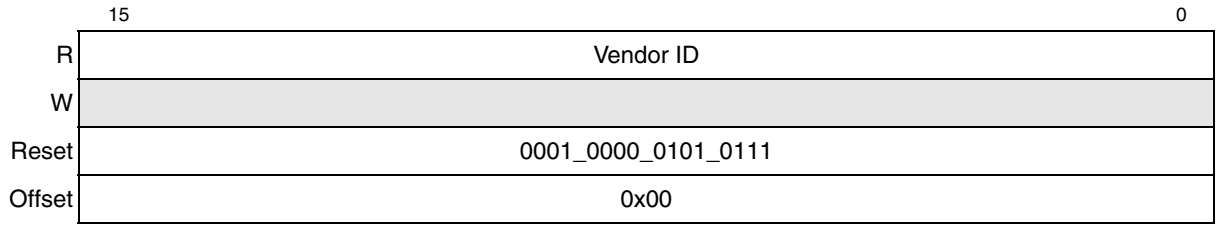


Figure 16-25. PCI Vendor ID Register

Table 16-24 describes PCI vendor ID register fields.

Table 16-24. PCI Vendor ID Register Field Descriptions

Bits	Name	Description
15–0	Vendor ID	0x1057 (Freescale)

16.3.2.2 PCI Device ID Register—Offset 0x02

The PCI device ID register, shown in Figure 16-26, is used to identify the device.

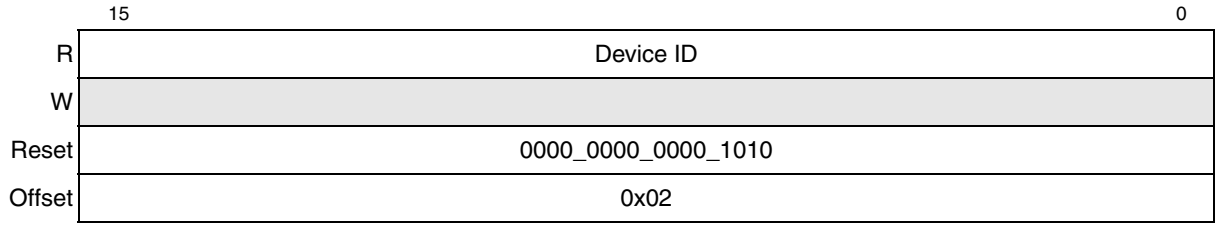


Figure 16-26. PCI Device ID Register

Table 16-25. PCI Device ID Register Field Descriptions

Bits	Name	Description
15–0	Device ID	0x000A: MPC8555E 0x000C: MPC8541

16.3.2.3 PCI Bus Command Register—Offset 0x04

The 2-byte PCI bus command register provides control over the ability to generate and respond to PCI cycles. Table 16-26 describes the bits of the PCI bus command register.

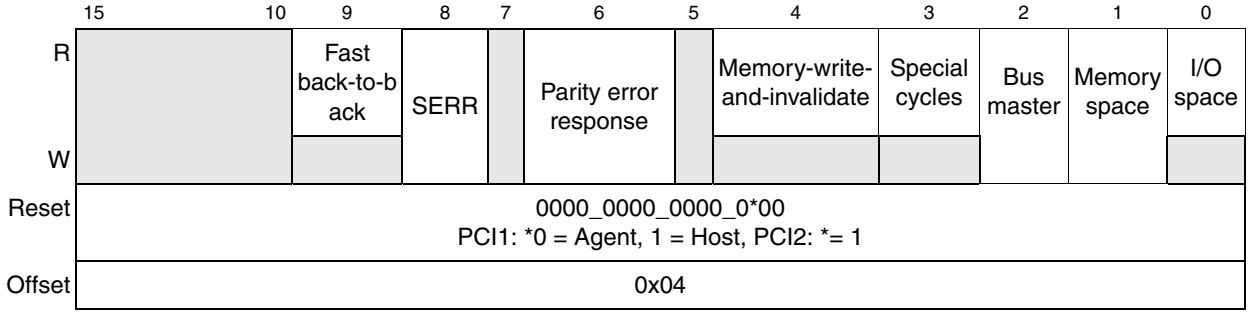


Figure 16-27. PCI Bus Command Register

Table 16-26. PCI Bus Command Register Field Descriptions

Bits	Name	Description
15–10	—	Reserved
9	Fast back-to-back	Hard-wired to 0, indicating that this PCI controller (as a master) does not run fast back-to-back transactions
8	SERR	Controls the $\overline{\text{PCI}n\_\text{SERR}}$ driver of this PCI controller. This bit (and bit 6) must be set to report address parity errors. 0 Disables the $\overline{\text{PCI}n\_\text{SERR}}$ driver 1 Enables the $\overline{\text{PCI}n\_\text{SERR}}$ driver
7	—	Reserved

Table 16-26. PCI Bus Command Register Field Descriptions (continued)

Bits	Name	Description
6	Parity error response	Controls whether this PCI controller responds to parity errors 0 Parity errors are ignored and normal operation continues. 1 Parity errors cause the appropriate bit in the PCI status register to be set. However, note that errors are reported based on the values set in the PCI error enable and detection registers.
5	—	Reserved
4	Memory-write-and-invalidate	Hard-wired to 0, indicating that this PCI controller, acting as a master, can not generate the memory-write-and-invalidate command.
3	Special-cycles	Hard-wired to 0, indicating that this PCI controller (as a target) ignores all special-cycle commands.
2	Bus master	Indicates whether this PCI controller is configured as a master. For PCI1, the reset state of this bit depends on whether PCI1 is configured in host mode (bus master = 1) or agent mode (bus master = 0). 0 Disables the ability to generate PCI accesses 1 Enables this PCI controller to behave as a PCI bus master Note that the bus master bit in the PCI bus command register should be set before attempting an outbound configuration access.
1	Memory space	Controls whether this PCI controller (as a target) responds to memory accesses 0 This PCI controller does not respond to PCI memory space accesses. 1 This PCI controller (as a target) responds to PCI memory space accesses.
0	I/O space	Hard-wired to 0, indicating that this PCI controller (as a target) does not respond to PCI I/O space accesses

16.3.2.4 PCI Bus Status Register—Offset 0x06

The 2-byte PCI bus status register is used to record status information for PCI bus bus-related events. The definition of each bit is given in Table 16-27. Only 2-byte accesses to address offset 0x06 are allowed.

Note that some errors are reported in two bits—one in the PCI bus status register and another in the PCI error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur. Refer to Table 16-52 for PCI mode error actions. Likewise, some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI error enable register (ERR_EN).

Reads to this register behave normally. Writes are slightly different in that bits can be cleared, but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 14 and not affect any other bits in the register, write the value 0b0100_0000_0000_0000 to the register.

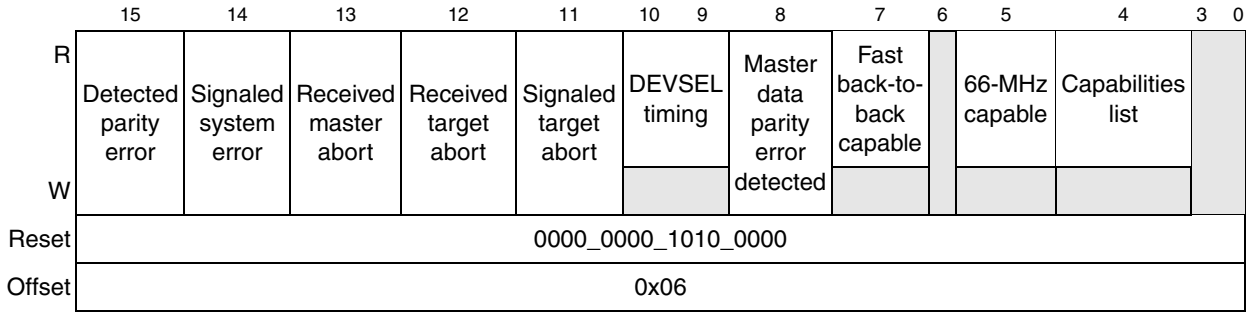


Figure 16-28. PCI Bus Status Register

Table 16-27. PCI Bus Status Register Field Descriptions

Bits	Name	Description
15	Detected parity error	Set whenever this PCI controller detects a PCI parity error, even if parity error handling is disabled (as controlled by bit 6 in the PCI bus command register
14	Signaled system error	Set whenever this PCI controller asserts $\overline{\text{PCI}n\_\text{SERR}}$
13	Received master-abort	Set whenever this PCI controller, acting as the PCI master, terminates a transaction (except for a special-cycle) using master-abort
12	Received target-abort	Set whenever a PCI transaction initiated by this PCI controller (excluding a special-cycle) is terminated by a target-abort
11	Signaled target-abort	Set whenever this PCI controller, acting as the PCI target, issues a target-abort to a PCI master
10–9	DEVSEL timing	Hard-wired to 0b00, indicating that this PCI controller uses fast device select timing
8	Master data parity error detected	Set upon detecting a data parity error. Three conditions must be met for this bit to be set: • This PCI controller detected a parity error. • This PCI controller was acting as the bus master for the operation in which the error occurred. • Bit 6 in the PCI bus command register was set.
7	Fast back-to-back capable	Hard-wired to 1, indicating that this PCI controller (as a target) is capable of accepting fast back-to-back transactions
6	—	Reserved
5	66-MHz capable	Read-only bit indicates that this PCI controller is capable of 66 MHz PCI bus operation
4	Capabilities list	Hard-wired to 0
3–0	—	Reserved

16.3.2.5 PCI Revision ID Register—Offset 0x08

The PCI Revision ID register is used to identify the revision of the part.

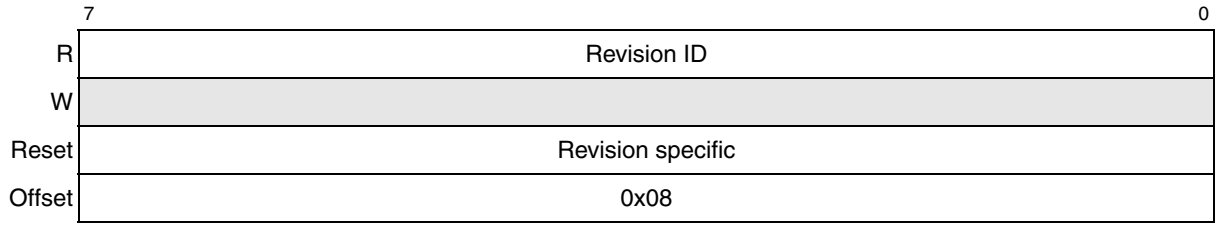


Figure 16-29. PCI Revision ID Register

Table 16-28. PCI Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific

16.3.2.6 PCI Bus Programming Interface Register—Offset 0x09

Table 16-29 describes the PCI bus programming interface register (PIR).

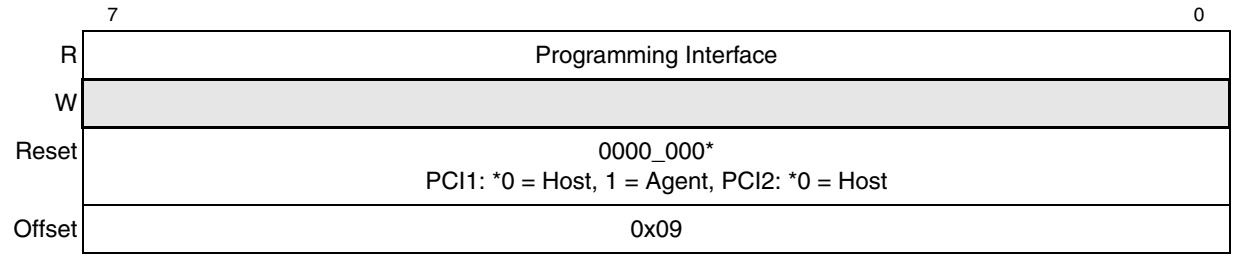


Figure 16-30. PCI Bus Programming Interface Register

Table 16-29. PCI Bus Programming Interface Register Field Descriptions

Bits	Name	Description
7–0	Programming Interface	0x00 When the PCI controller is configured as host bridge 0x01 When the PCI controller is configured as an agent device

16.3.2.7 PCI Subclass Code Register—Offset 0x0A

Table 16-31 describes the PCI subclass code register (PSCR).

	7	0
R	Subclass Code	
W		
Reset	0010_0000	
Offset	0x0A	

Figure 16-31. PCI Subclass Code Register

Table 16-30. PCI Subclass Code Register Field Descriptions

Bits	Name	Description
7–0	Subclass Code	PowerPC—0x20

16.3.2.8 PCI Bus Base Class Code Register—Offset 0x0B

Table 16-31 describes the PCI bus base class code register (PBCCR).

	7	0
R	Base Class Code	
W		
Reset	0000_1011	
Offset	0x0B	

Figure 16-32. PCI Bus Base Class Code Register

Table 16-31. PCI Bus Base Class Code Register Field Descriptions

Bits	Name	Description
7–0	Base Class Code	Processor—0x0B

16.3.2.9 PCI Bus Cache Line Size Register—Offset 0x0C

Table 16-32 describes the PCI bus cache line size register (PCLSR).

	7	0
R W	Cache Line Size	
Reset	0000_0000	
Offset	0x0C	

Figure 16-33. PCI Bus Cache Line Size Register

Table 16-32. PCI Bus Cache Line Size Register Field Descriptions

Bits	Name	Description
7–0	Cache Line Size	Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). PCLSR is read-write, however for PCI operation an attempt to program this register to any value other than 0x8 results in clearing it.

16.3.2.10 PCI Bus Latency Timer Register—0x0D

Table 16-33 describes the PCI latency timer register (PLTR).

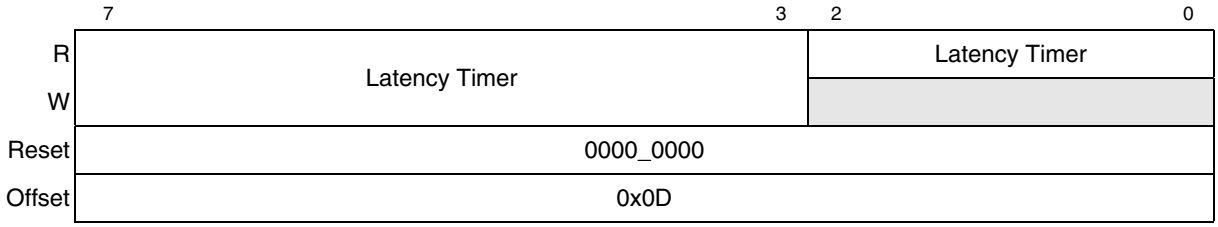


Figure 16-34. PCI Bus Latency Timer Register

Table 16-33. PCI Bus Latency Timer Register Field Descriptions

Bits	Name	Description
7–3	Latency Timer	The maximum number of PCI clocks that the device, when mastering a transaction, holds the bus after PCI bus grant has been negated The value is in PCI clocks. The PCI 2.2 specification gives rules by which the PCI bus interface unit completes transactions when the timer has expired.
2–0	Latency Timer	Read-only bits. The minimum latency timer value when set is 8 PCI clocks.

16.3.2.11 PCI Base Address Registers

A PCI base address register points to the beginnings of each address range to which the device responds by asserting $\overline{\text{PCI}n\text{\_DEVSEL}}$. The base address register (BAR) at offset 0x10 is a fixed 1-Mbyte window that is automatically translated to the local configuration, control, and status registers address space.

The other base address registers are aliases (with differing format) of the PCI inbound ATMU windows; see Section 16.3.1.3, “PCI ATMU Inbound Registers.” The 32-bit base address register at offset 0x14 corresponds to inbound ATMU window 1; the 64-bit base address registers at offsets 0x18 and 0x20 correspond to inbound ATMU windows 2 and 3. If one of these registers is written, the corresponding ATMU register is also updated; if a PCI inbound ATMU register is written, the corresponding BAR is also updated. If one of these registers is read, the corresponding size of ATMU is returned on the PCI bus providing valid window size in the Inbound ATMU window attributes register.

Note that PCSRBAR cannot be updated through the inbound ATMU registers.

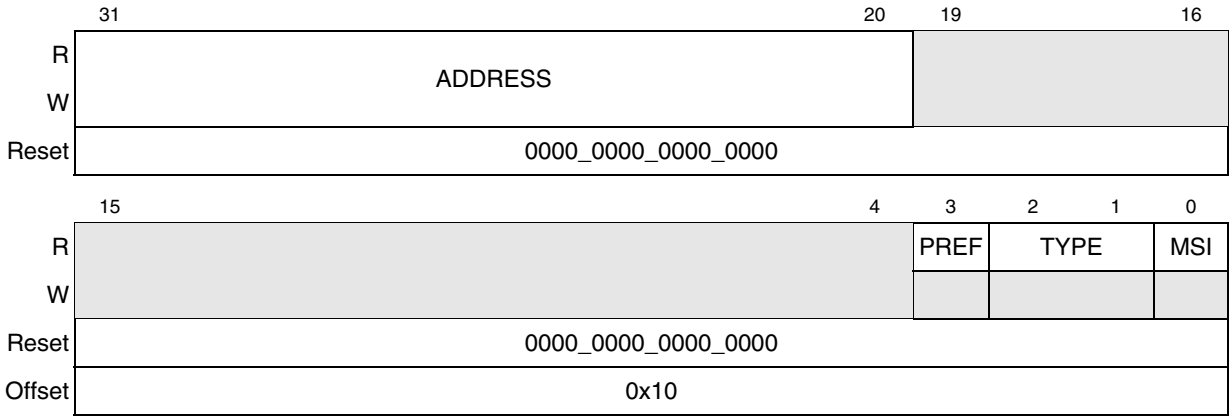


Figure 16-35. PCI Configuration and Status Register Base Address Register (PCSRBAR)

Table 16-34. PCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration/run-time window resides at. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00—Locate anywhere in 32-bit address space
0	MSI	Memory space indicator

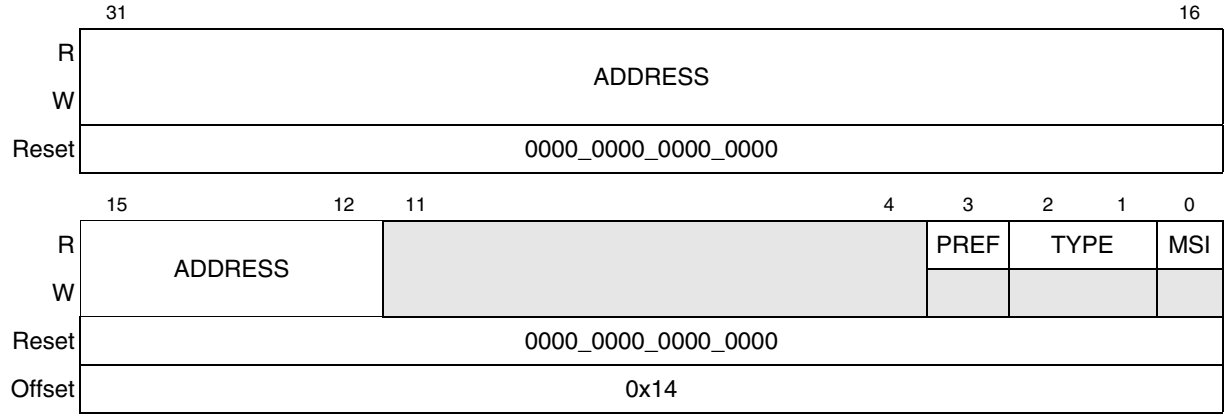


Figure 16-36. 32–Bit Memory Base Address Register

Table 16-35. 32-Bit Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows.
11–4	—	Reserved. The device allows a 4-Kbyte window minimum.

Table 16-35. 32-Bit Memory Base Address Register Field Descriptions (continued)

Bits	Name	Description
3	PREF	Prefetchable
2–1	TYPE	Type. 00 - Locate anywhere in 32-bit address space
0	MSI	Memory space indicator

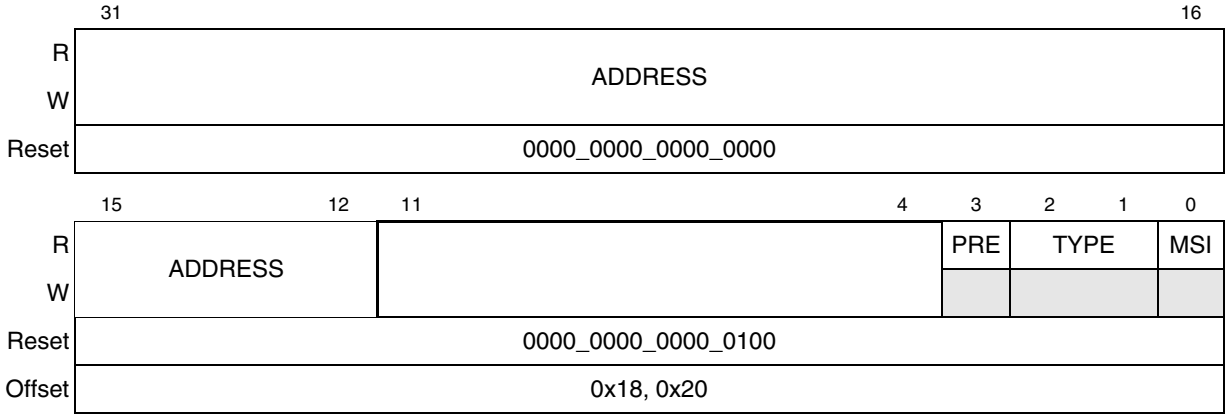


Figure 16-37. 64-Bit Low Memory Base Address Register

Table 16-36. 64-Bit Low Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows.
11–4	—	Reserved. The device allows a 4-Kbyte window minimum.
3	PREF	Prefetchable
2–1	TYPE	Type. 10 - Locate anywhere in 64-bit address space
0	MSI	Memory space indicator

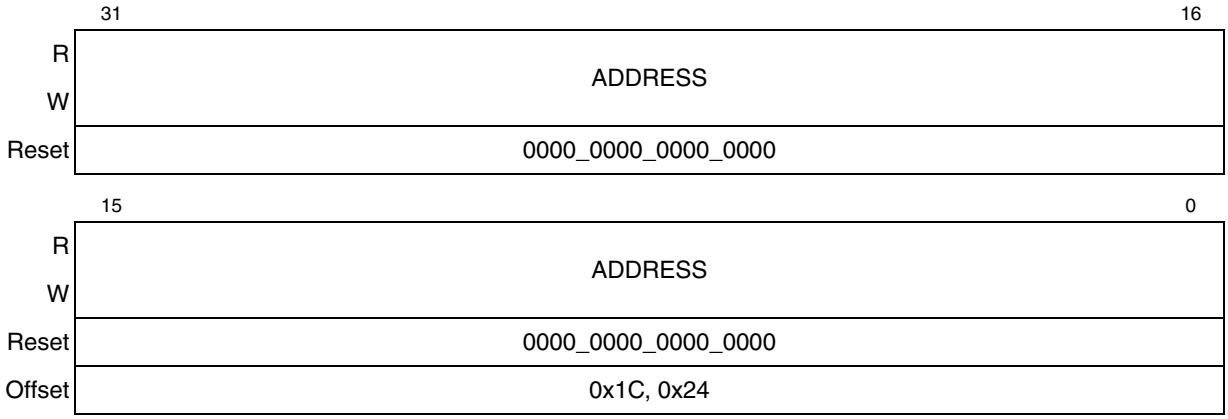


Figure 16-38. 64-Bit High Memory Base Address Register

Table 16-37. Bit Setting for 64-Bit High Memory Base Address Register

Bits	Name	Description
31–0	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows. If no access to local memory is to be permitted by external masters then all bits are programmed.

16.3.2.12 PCI Subsystem Vendor ID Register

The PCI subsystem vendor ID register is used to identify the subsystem.

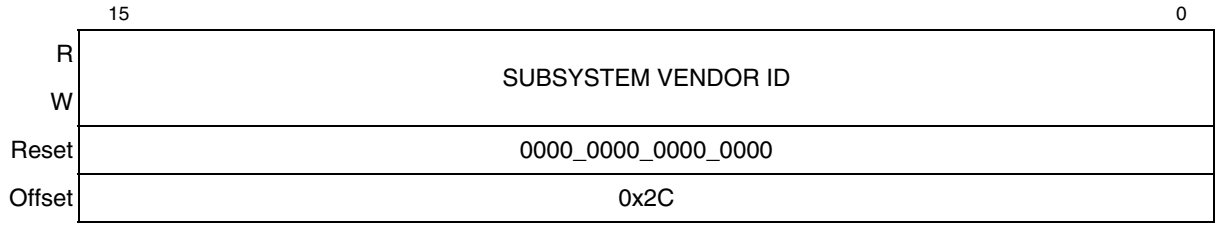


Figure 16-39. PCI Subsystem Vendor ID Register

Table 16-38. PCI Subsystem Vendor ID Register Field Descriptions

Bits	Name	Description
15–0	Subsystem Vendor ID	0x0000

16.3.2.13 PCI Subsystem ID Register

The PCI subsystem ID register is used to identify the subsystem.

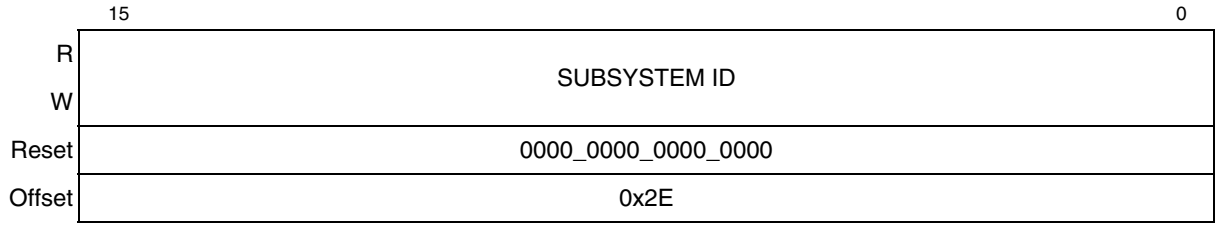


Figure 16-40. PCI Subsystem ID Register

Table 16-39. PCI Subsystem ID Register Field Descriptions

Bits	Name	Description
15–0	Subsystem ID	0x0000

16.3.2.14 PCI Bus Capabilities Pointer Register

The PCI bus capabilities pointer identifies additional functionality supported by the device.

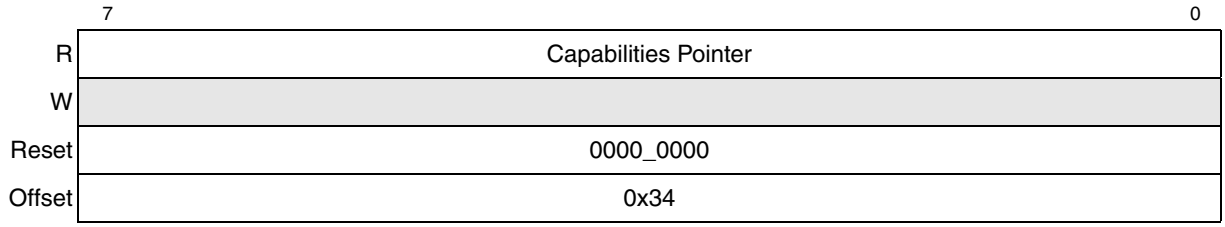


Figure 16-41. PCI Bus Capabilities Pointer Register

Table 16-40. PCI Bus Capabilities Pointer Register Field Descriptions

Bits	Name	Description
7–0	Capabilities Pointer	No additional capabilities

16.3.2.15 PCI Bus Interrupt Line Register

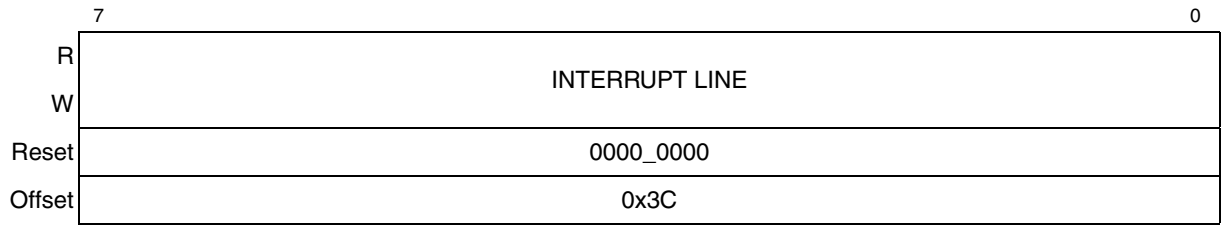


Figure 16-42. PCI Bus Interrupt Line Register

Table 16-41. PCI Bus Interrupt Line Register Field Descriptions

Bits	Name	Description
7–0	Interrupt Line	This register is used to communicate interrupt line routing information.

16.3.2.16 PCI Bus Interrupt Pin Register

The MPC8555E has 12 general purpose interrupt request lines (IRQ[0:11]) and an interrupt output, $\overline{\text{IRQ\_OUT}}$ (active low, level sensitive), to which all external and most internal interrupt sources (including PCI) can be routed. $\overline{\text{IRQ\_OUT}}$ is mapped to PCI_INTA as a default. Note that this device does not respond to INTACK or special cycle commands on the PCI interfaces.

	7	0
R	Interrupt Pin	
W		
Reset	0000_0001	
Offset	0x3D	

Figure 16-43. PCI Bus Interrupt Pin Register

Table 16-42. PCI Bus Interrupt Pin Register Field Descriptions

Bits	Name	Description
7–0	Interrupt Pin	PCI_INTA pin selected

16.3.2.17 PCI Bus Minimum Grant (MIN GNT) Register

	7	0
R	MINGNT	
W		
Reset	0000_0000	
Offset	0x3E	

Figure 16-44. PCI Bus Minimum Grant Register

Table 16-43. PCI Bus Minimum Grant Register Field Descriptions

Bits	Name	Description
7–0	MINGNT	Specifies the length of the device's burst period (0x00 indicates that this PCI controller has no major requirements for the settings of latency timers)

16.3.2.18 PCI Bus Maximum Latency (MAX LAT) Register

	7	0
R	MAXLAT	
W		
Reset	0000_0000	
Offset	0x3F	

Figure 16-45. PCI Bus Maximum Latency Register

Table 16-44. PCI Bus Maximum Latency Register Field Descriptions

Bits	Name	Description
7–0	MAXLAT	Specifies how often the device needs to gain access to the PCI bus (0x00 indicates that this PCI controller has no major requirements for the settings of latency timers)

16.3.2.19 PCI Bus Function Register (PBFR)

The 2-byte PCI bus function register is used to determine different features how the PCI interface in bus 0 is configured. This register is at PCI configuration space at offset 0x44.

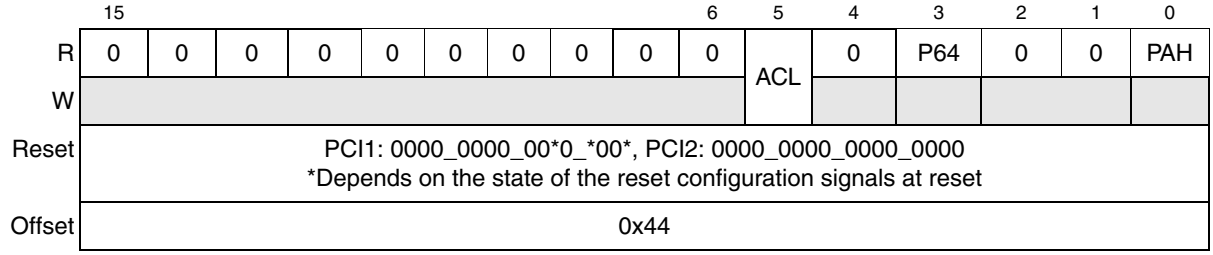


Figure 16-46. PCI Bus Function Register

Table 16-45. PCI Bus Function Register Field Descriptions

Bits	Name	Description
15–6	—	Reserved
5	ACL	Agent configuration lock. Indicates to an external host whether the local processor is doing internal configuration and must be explicitly set and cleared by the local processor during this time. ACL is set during reset if the LA27 (cfg_cpu_boot) input selects the CPU as the configuration owner. This bit is only meaningful in agent mode. 0 PCI interface allows incoming PCI configuration cycles 1 PCI interface retries all incoming PCI configuration cycles
4	—	Reserved
3	P64	PCI 64-bit configuration. Read-only. Indicates the reset value of the PCI 64-bit configuration signal, $\overline{\text{PCI1\_REQ64}}$. 0 64-bit interface functions as a 32-bit interface 1 64-bit interface functions as a 64-bit interface
2–1	—	Reserved
0	PAH	PCI agent/host. Read-only. Indicates the reset value of the PCI host/agent configuration signal, $\overline{\text{LWE2}}$. 0 PCI interface is in host mode 1 PCI interface is in agent mode

16.3.2.20 PCI Bus Arbiter Configuration Register (PBACR)

The PCI bus arbiter configuration register is used to determine the configuration of the PCI bus arbiter.

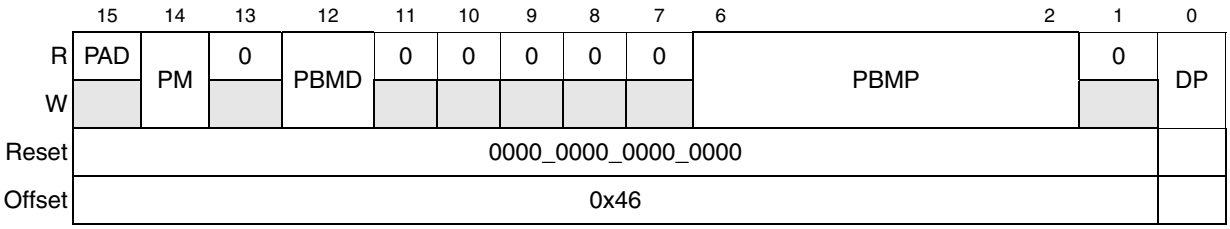


Figure 16-47. PCI Bus Arbiter Configuration Register

Table 16-46. PCI Bus Arbiter Configuration Register Field Descriptions

Bits	Name	Description
15	PAD	PCI arbiter disable. Determines if the device is the PCI arbiter on the PCI bus or not. The reset state is determined by the inverse of the $\overline{\text{PCI}n\text{\_GNT2}}$ configuration input signal when reset is released. 0 Device is the PCI arbiter 1 Device is not the PCI arbiter. Device presents its request on $\overline{\text{PCI}n\text{\_REQ0}}$ to the external arbiter and receives its grant on $\overline{\text{PCI}n\text{\_GNT0}}$
14	PM	Parking mode. controls which device receives the bus grant when there are no outstanding bus requests and the bus is idle. 0 The bus is parked on the last device to use the bus. 1 The bus is parked on the device.
13	—	Reserved
12	PBMD	PCI broken master disable. Determines if the device ignores the bus requests of an initiator that requests the bus for an excessive period without using the bus 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 else its request is subsequently ignored. 1 No requests are ignored.
11–7	—	Reserved
6–2	PBMP	PCI bus master priorities. Determines arbitration priority given to different masters on the PCI bus. Bit 6 corresponds to the priority of the master sourcing $\overline{\text{PCI}n\text{\_REQ0}}$; bit 2 corresponds to the priority of the master sourcing $\overline{\text{PCI}n\text{\_REQ4}}$. 0 Master *n* is low priority 1 Master *n* is high priority
1	—	Reserved
0	DP	Device priority. Determines this device's arbitration priority 0 Device is low priority 1 Device is high priority

16.4 Functional Description

This section describes the functionality of the PCI interface.

16.4.1 PCI Bus Arbitration

PCI bus arbitration is access-based. Bus masters must arbitrate for each access performed on the bus. The PCI bus uses a central arbitration scheme where each master has its own unique request ($\overline{\text{REQ}}$) output and grant ($\overline{\text{GNT}}$) 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 due to arbitration (except when the bus is idle).

The MPC8555E provides bus arbitration logic for its master interface and up to five other external PCI bus masters. The on-chip PCI arbiter is independent of host or agent mode. The on-chip PCI arbiter functions in both host and agent modes, or it can be disabled to allow for an external PCI arbiter.

A configuration signal ($\overline{\text{PCI}n\text{\_GNT2}}$) sampled at the negation of $\overline{\text{HRESET}}$ determines if the on-chip PCI arbiter is enabled (high) or disabled (low). The state of the reset signal is reflected in bit 15 (read-only) of

the PCI bus arbitration control register (PBACR[PAD]). Note that PAD is the inverse of the arbiter configuration signal; that is, when PAD = 0 the arbiter is enabled, and when PAD = 1 the arbiter is disabled. See Chapter 4, "Reset, Clocking, and Initialization," for more information on the reset configuration signals.

If the on-chip PCI arbiter is enabled, a request-grant pair of signals is provided for each external master ($\overline{\text{PCI}n\text{\_REQ}}$[0:4] and $\overline{\text{PCI}n\text{\_GNT}}$[0:4]). In addition, there is an internal request/grant pair for the internal master state machine of the MPC8555E that governs internal accesses to the PCI interface. If the on-chip PCI arbiter is disabled, the MPC8555E uses the $\overline{\text{PCI}n\text{\_REQ0}}$ signal as an output to issue its request to the external arbiter and uses the $\overline{\text{PCI}n\text{\_GNT0}}$ signal as an input to receive its grant from the external arbiter.

The following sections describe the operation of the on-chip PCI arbiter that arbitrates between external PCI masters and the internal PCI bus master of the MPC8555E.

16.4.1.1 PCI Bus Arbiter Operation

The on-chip PCI arbiter uses a programmable two-level, round-robin arbitration algorithm. Each of the five external masters, plus the MPC8555E, can be programmed for two priority levels, high or low, using the appropriate bits in the PBACR. Within each priority group, the PCI bus grant is asserted to the next requesting device in numerical order, with the MPC8555E positioned before device 0.

Conceptually, the lowest priority device is the master that is currently using the bus, and the highest priority device is the device that follows the current master in numerical order and group priority. This is considered to be a fair algorithm, since a single device cannot prevent other devices from having access to the bus; it automatically becomes the lowest priority device as soon as it begins to use the bus. If a master is not requesting the bus, then its transaction slot is given to the next requesting device within its priority group.

A grant is awarded to the highest priority requesting device as soon as the current master begins a transaction; however, the granted device must wait until the bus is relinquished by the current master before initiating a transaction.

The grant given to a particular device may be removed and awarded to another higher priority device, whenever the higher priority device asserts its request. If the bus is idle when a device requests the bus, then the arbiter withholds the grant for one clock cycle. The arbiter re-evaluates the priorities of all requesting devices and grants the bus to the highest priority device in the following clock cycle. This allows a turnaround clock when a higher priority device is using address stepping or when the bus is parked.

The low-priority group collectively has one bus transaction request slot in the high-priority group. For N high-priority devices and M low-priority devices, each high-priority device is guaranteed at least 1 of N+1 bus transactions and each low-priority device is guaranteed at least 1 of (N+1) × M bus transactions, with one low-priority device receiving the grant in 1 of N+1 bus transactions. If all devices are programmed to the same priority level, or if the low-priority group has only one device, the algorithm defaults to give each device an equal number of bus grants in round-robin sequence.

For the example in Figure 16-48, assume that several devices are requesting the bus. If two masters are in the high-priority group and three are in the low-priority group, each high-priority master is guaranteed at

least one out of three transaction slots and each low-priority master is guaranteed one out of nine transaction slots.

In Figure 16-48, the grant sequence (with all devices, except device 4 requesting the bus and device 3 being the current master) is 0, 2, MPC8555E, 0, 2, 1, 0, 2, 3, …, and repeating. If device 2 is not requesting the bus, the grant sequence is 0, MPC8555E, 0, 1, 0, 3, …, and repeating. If device 2 requests the bus when device 0 is conducting a transaction and the MPC8555E has the next grant, the MPC8555E has its grant removed and device 2 is awarded the grant since device 2 is higher priority than the MPC8555E when device 0 has the bus.

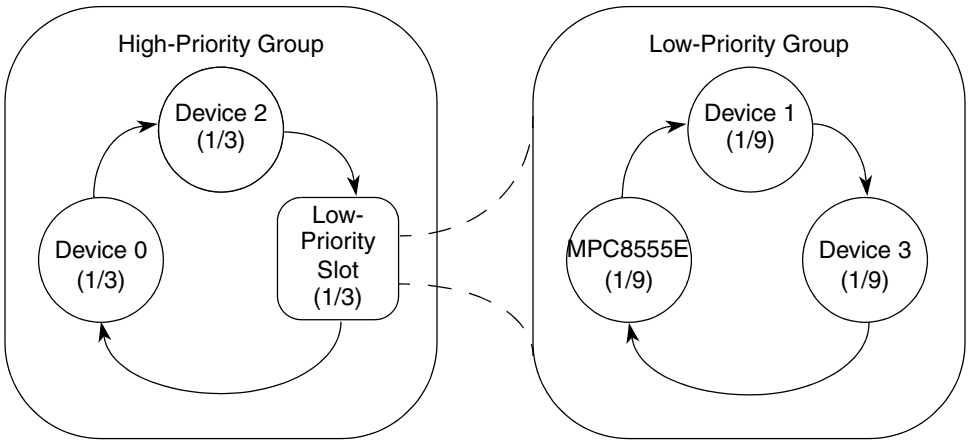


Figure 16-48. PCI Arbitration Example

16.4.1.2 PCI Bus Parking

When no device is using or requesting the bus, the PCI arbiter grants the bus to a selected device. This is known as parking the bus on the selected device. The selected device is required to drive the PCI*n*_AD[31:0], PCI*n*_C/$\overline{\text{BE}}$[3:0], and the PCI parity signals to a stable value, preventing these signals from floating.

The parking mode parameter (PBACR[PM]) determines which device the arbiter selects for parking the PCI bus. If PBACR[PM] = 0 (or if the bus is not idle), then the bus is parked on the last master to use the bus. If the bus is idle and PBACR[PM] = 1, the bus is parked on the MPC8555E.

16.4.1.3 Broken Master Lock-Out

The PCI bus arbiter on the MPC8555E has a feature that allows it to lock out any masters that are broken or ill-behaved. The broken master feature is controlled by programming bit 12 of the PCI bus arbitration control register (0 = enabled, 1 = disabled).

When the broken master feature is enabled, a granted device that does not assert $\overline{\text{PCI}n\text{\_FRAME}}$ 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.

16.4.1.4 Power-Saving Modes and the PCI Arbiter

In the sleep power-saving mode, the clock signal driving SYSCLK can be disabled. If the clock is disabled, the arbitration logic is not able to perform its function. System programmers must park the bus with a device that can sustain the PCI*n*_AD[31:0], PCI*n*_C/$\overline{\text{BE}}$[3:0], and parity signals prior to disabling the SYSCLK signal. If the bus is parked on the MPC8555E when its clocks are stopped, the MPC8555E sustains the PCI*n*_AD[31:0], PCI*n*_C/$\overline{\text{BE}}$[3:0], and parity signals in their prior states. In this situation, the only way for another agent to use the PCI bus is by waking the MPC8555E. In nap and doze power-saving modes, the arbiter continues to operate allowing other PCI devices to run transactions.

16.4.2 PCI Bus Protocol

This section provides a general description of the PCI bus protocol. Specific PCI bus transactions are described in Section 16.4.2.7, "PCI Bus Transactions." Refer to Figure 16-49, Figure 16-50, Figure 16-51, and Figure 16-52 for examples of the transfer-control mechanisms described in this section.

All signals are sampled on the rising edge of the PCI bus clock (SYSCLK). Each signal has a setup and hold aperture with respect to the rising clock edge in which transitions are not allowed. Outside this aperture, signal values or transitions have no significance. See the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for specific setup and hold times.

16.4.2.1 Basic Transfer Control

The basic PCI bus transfer mechanism is a burst. A burst is composed of an address phase followed by one or more data phases. Fundamentally, all PCI data transfers are controlled by three signals—$\overline{\text{PCI}n\text{\_FRAME}}$ (frame), $\overline{\text{PCI}n\text{\_IRDY}}$ (initiator ready), and $\overline{\text{PCI}n\text{\_TRDY}}$ (target ready). An initiator asserts $\overline{\text{PCI}n\text{\_FRAME}}$ to indicate the beginning of a PCI bus transaction and negates $\overline{\text{PCI}n\text{\_FRAME}}$ to indicate the end of a PCI bus transaction. An initiator negates $\overline{\text{PCI}n\text{\_IRDY}}$ to force wait cycles. A target negates $\overline{\text{PCI}n\text{\_TRDY}}$ to force wait cycles.

The PCI bus is considered idle when both $\overline{\text{PCI}n\text{\_FRAME}}$ and $\overline{\text{PCI}n\text{\_IRDY}}$ are negated. The first clock cycle in which $\overline{\text{PCI}n\text{\_FRAME}}$ is asserted indicates the beginning of the address phase. The address and bus command code are transferred in that first cycle. The next cycle begins the first of one or more data phases. Data is transferred between initiator and target in each cycle that both $\overline{\text{PCI}n\text{\_IRDY}}$ and $\overline{\text{PCI}n\text{\_TRDY}}$ are asserted. Wait cycles may be inserted in a data phase by the initiator (by negating $\overline{\text{PCI}n\text{\_IRDY}}$) or by the target (by negating $\overline{\text{PCI}n\text{\_TRDY}}$).

Once an initiator has asserted $\overline{\text{PCI}n\text{\_IRDY}}$, it cannot change $\overline{\text{PCI}n\text{\_IRDY}}$ or $\overline{\text{PCI}n\text{\_FRAME}}$ until the current data phase completes regardless of the state of $\overline{\text{PCI}n\text{\_TRDY}}$. Once a target has asserted $\overline{\text{PCI}n\text{\_TRDY}}$ or $\overline{\text{PCI}n\text{\_STOP}}$, it cannot change $\overline{\text{PCI}n\text{\_DEVSEL}}$, $\overline{\text{PCI}n\text{\_TRDY}}$, or $\overline{\text{PCI}n\text{\_STOP}}$ until the current data phase completes. In simpler terms, once an initiator or target has committed to the data transfer, it cannot change its mind.

When the initiator intends to complete only one more data transfer (which could be immediately after the address phase), $\overline{\text{PCI}n\text{\_FRAME}}$ is negated and $\overline{\text{PCI}n\text{\_IRDY}}$ is asserted (or kept asserted), indicating the initiator is ready. After the target indicates the final data transfer (by asserting $\overline{\text{PCI}n\text{\_TRDY}}$), the PCI bus may return to the idle state (both $\overline{\text{PCI}n\text{\_FRAME}}$ and $\overline{\text{PCI}n\text{\_IRDY}}$ are negated) unless a fast back-to-back

transaction is in progress. In the case of a fast back-to-back transaction, an address phase immediately follows the last data phase.

16.4.2.2 PCI Bus Commands

A PCI bus command is encoded in the PCI*n*_C/$\overline{\text{BE}}$[3:0] signals during the address phase of a PCI transaction. The bus command indicates to the target the type of transaction the initiator is requesting. Table 16-47 describes the PCI bus commands implemented by the MPC8555E.

Table 16-47. PCI Bus Commands

PCI*n*_C/$\overline{\text{BE}}$[3:0]	PCI Bus Command	MPC8555E Supports as an Initiator	MPC8555E Supports as a Target	Definition
0000	Interrupt-acknowledge	Yes	No	A read (implicitly addressing the system interrupt controller). Only one device on the PCI bus should respond to this command; others ignore it. See Section 16.4.2.12.1, "Interrupt-Acknowledge Transactions," for more information.
0001	Special cycle	Yes	No	Provides a way to broadcast select messages to all devices on the PCI bus. See Section 16.4.2.12.2, "Special-Cycle Transactions," for more information.
0010	I/O-read	Yes	No	Accesses agents mapped into the PCI I/O space
0011	I/O-write	Yes	No	Accesses agents mapped into the PCI I/O space
0100	Reserved[1]	No	No	—
0101	Reserved[1]	No	No	—
0110	Memory-read	Yes	Yes	Accesses either local memory or agents mapped into PCI memory space, depending on the address. When a PCI master issues this command to local memory, the MPC8555E (the target) fetches data from the requested address to the end of the cache line (32 bytes) from local memory, even though all of the data may not be requested by (or sent to) the initiator.
0111	Memory-write	Yes	Yes	Accesses either local memory or agents mapped into PCI memory space, depending on the address
1000	Reserved[1]	No	No	—
1001	Reserved[1]	No	No	—
1010	Configuration-read	Yes	Agent mode only	Accesses the 256-byte configuration space of a PCI agent. A specific agent is selected when its IDSEL signal is asserted during the address phase. See Section 16.4.2.11, "Configuration Cycles," for details.
1011	Configuration-write	Yes	Agent mode only	
1100	Memory-read-multiple	Yes	Yes	Similar to the memory-read command, but also causes a prefetch of the next cache line (32 bytes)
1101	Dual-address-cycle	Yes	Yes	Used to transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices

Table 16-47. PCI Bus Commands (continued)

PCI*n*_C/$\overline{BE}$[3:0]	PCI Bus Command	MPC8555E Supports as an Initiator	MPC8555E Supports as a Target	Definition
1110	Memory-read-line	Yes	Yes	Indicates that an initiator is requesting the transfer of an entire cache line. This occurs only when the processor is performing a burst read. Note that these processors perform burst reads only when the appropriate cache is enabled and the transaction is not cache-inhibited.
1111	Memory-write-and-invalidate	No	Yes	Indicates that an initiator is transferring an entire cache line; if this data is in any cacheable memory, that cache line needs to be invalidated.

[1] Reserved command encodings are reserved for future use. The MPC8555E does not respond to these commands.

16.4.2.3 Addressing

PCI defines three physical address spaces—PCI memory space, PCI I/O space, and PCI configuration space. Access to the PCI memory and I/O space is straightforward, although one must take into account the local memory access window and address translation being used. The address translation registers are described in Section 16.3.1, "PCI Memory Mapped Registers." Access to the PCI configuration space is described in Section 16.4.2.11, "Configuration Cycles."

Address decoding on the PCI bus is performed by every device for every PCI transaction. Each agent is responsible for decoding its own address. PCI supports two types of address decoding—positive decoding and subtractive decoding. For positive decoding, each device looks for accesses in the address range that the device has been assigned. For subtractive decoding, one device on the bus looks for accesses that no other device has claimed. See Section 16.4.2.4, "Device Selection," for information about claiming transactions.

The information contained in the two low-order address bits (PCI*n*_AD[1:0]) varies by the address space (memory, I/O, or configuration). Regardless of the encoding scheme, the two low-order address bits are always included in parity calculations.

16.4.2.3.1 Memory Space Addressing

For memory accesses, PCI defines two types of burst ordering controlled by the two low-order bits of the address—linear incrementing (PCI*n*_AD[1:0] = 0b00) and cache wrap mode (PCI*n*_AD[1:0] = 0b10), as shown in Table 16-48. The other two PCI*n*_AD[1:0] possibilities (0b01 and 0b11) are reserved. As an initiator, the MPC8555E always encodes PCI*n*_AD[1:0] = 00 for PCI memory space accesses. As a target, the MPC8555E executes a target disconnect after the first data phase completes if PCI*n*_AD[1:0] = 01 or PCI*n*_AD[1:0] = 0b11 during the address phase of a local memory access. See Section 16.4.2.8.2, "Target-Initiated Termination," for more information on target disconnect conditions.

Table 16-48. Supported Combinations of PCI*n*_AD[1:0]

PCI*n*_AD[1:0]		MPC8555E as Target		MPC8555E as Initiator	
		Read	Write	Read	Write
00	Linear	√	√	√	√
01	Reserved	TD	TD	—	—
10	Cache wrap	√	TD	—	—
11	Reserved	TD	TD	—	—

For linear incrementing mode, the memory address is encoded/decoded using PCI*n*_AD[31:2] or PCI1_AD[63:2] in 64-bit mode. Thereafter, the address is incremented by 4 bytes after each data phase completes until the transaction is terminated or completed (a 4-byte data width per data phase is implied). Note that the two low-order bits on the address bus are included in all parity calculations.

For cache wrap mode (PCI*n*_AD[1:0] = 0b10) reads, the critical memory address is decoded using PCI*n*_AD[31:2] or PCI1_AD[63:2] in 64-bit mode. The address is incremented by 4 bytes after each data phase completes until the end of the cache line is reached. For cache-wrap reads, the address wraps to the beginning of the current cache line and continues incrementing until the entire cache line (32 bytes) is read. The MPC8555E does not support cache-wrap write operations and executes a target disconnect after the data phase for the end of the cache line completes for writes with PCI*n*_AD[1:0] = 0b10. That is, the MPC8555E does not wrap back to the beginning of the cache line. Note that the two low-order bits on the address bus are included in all parity calculations.

16.4.2.3.2 I/O Space Addressing

For PCI I/O accesses, 32 address signals (PCI*n*_AD[31:0]) are used to provide a byte address. After a target has claimed an I/O access, it must determine 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 of the target, the entire access cannot complete. In this case, the target does not transfer any data and terminates the transaction with a target-abort error. See Section 16.4.2.8.2, “Target-Initiated Termination,” for more information.

16.4.2.3.3 Configuration Space Addressing

PCI supports two types of configuration accesses that use different formats for the PCI*n*_AD[31:0] signals during the address phase. The two low-order bits of the address indicate the format used for the configuration address phase—type 0 (PCI*n*_AD[1:0] = 0b00) or type 1 (PCI*n*_AD[1:0] = 0b01). Both address formats identify a specific device and a specific configuration register for that device. See Section 16.4.2.11, “Configuration Cycles,” for descriptions of the two formats.

16.4.2.4 Device Selection

The $\overline{\text{PCI}n\text{\_DEVSEL}}$ signal is driven by the target of the current transaction. $\overline{\text{PCI}n\text{\_DEVSEL}}$ indicates to the other devices on the PCI bus that the target has decoded the address and claimed the transaction. $\overline{\text{PCI}n\text{\_DEVSEL}}$ may be driven one, two, or three clock cycles (fast, medium, or slow device select timing) following the address phase. Device select timing is encoded into the device’s PCI bus status register. If

no agent asserts PCI*n*_DEVSEL within three clock cycles of PCI*n*_FRAME, the agent responsible for subtractive decoding may claim the transaction by asserting PCI*n*_DEVSEL.

A target must assert PCI*n*_DEVSEL (claim the transaction) before or coincident with any other target response (assert PCI*n*_TRDY, PCI*n*_STOP, or data signals). In all cases except target-abort, once a target asserts PCI*n*_DEVSEL, it must not negate PCI*n*_DEVSEL until PCI*n*_FRAME is negated (with PCI*n*_IRDY asserted) and the last data phase has completed. For normal termination, negation of PCI*n*_DEVSEL coincides with the negation of PCI*n*_TRDY or PCI*n*_STOP.

If the first access maps into a target's address range, that target asserts PCI*n*_DEVSEL to claim the access. However, if the initiator attempts to continue the burst access across the resource boundary, then the target must issue a target disconnect.

The MPC8555E is hardwired for fast device select timing (PCI bus status register [10–9] = 0b00). Therefore, when the MPC8555E is the target of a transaction (local memory access or configuration register access), it asserts PCI*n*_DEVSEL one clock cycle following the address phase.

As an initiator, if the MPC8555E does not detect the assertion of PCI*n*_DEVSEL within four clock cycles after the address phase (that is, five clock cycles after it asserts PCI*n*_FRAME), it terminates the transaction with a master-abort termination; see Section 16.4.2.8.1, "Master-Initiated Termination."

16.4.2.5 Byte Alignment

The byte enable signals of the PCI bus (PCI*n*_C/BE, during a data phase) are used to determine which byte lanes carry meaningful data. The byte enable signals may enable different bytes for each of the data phases. The byte enables are valid on the edge of the clock that starts each data phase and stay valid for the entire data phase. Note that parity is calculated for all bytes regardless of the state of the byte enable signals. See Section 16.4.2.13.1, "PCI Parity," for more information.

If the MPC8555E, as a target, detects no byte enables asserted, it completes the current data phase with no permanent change. This implies that on a read transaction, the MPC8555E expects that the data is not changed, and on a write transaction, the data is not stored.

16.4.2.6 Bus Driving and Turnaround

To avoid contention, a turnaround cycle is required on all signals that may be driven by more than one agent. The turnaround cycle occurs at different times for different signals. The PCI*n*_IRDY, PCI*n*_TRDY, PCI*n*_DEVSEL, and PCI*n*_STOP signals use the address phase as their turnaround cycle. PCI*n*_FRAME, PCI*n*_C/BE[3:0], and PCI*n*_AD[31:0] (or PCI1_C/BE[7:0] and PCI1_AD[63:2] in 64-bit mode) signals use the idle cycle between transactions (when both PCI*n*_FRAME and PCI*n*_IRDY are negated) as their turnaround cycle. PCI*n*_PERR has a turnaround cycle on the fourth clock cycle after the last data phase.

The PCI address/data signals, PCI*n*_AD[31:0] or PCI1_AD[63:2] in 64-bit mode, are driven to a stable condition during every address/data phase. Even when the byte enables indicate that byte lanes carry meaningless data, the signals carry stable values. Parity is calculated on all bytes regardless of the byte enables. See Section 16.4.2.13.1, "PCI Parity," for more information.

16.4.2.7 PCI Bus Transactions

This section provides descriptions of the PCI bus transactions. All bus transactions follow the protocol as described in Section 16.4.2, "PCI Bus Protocol." Read and write transactions are similar for the memory and I/O spaces, so they are described as generic read transactions and generic write transactions.

The timing diagrams in this section show the relationship of significant signals involved in bus transactions. When a signal is drawn as a solid line, it is actively being driven by the current master or target. When a signal is drawn as a dashed line, no agent is actively driving it. High-impedance signals are indicated to have indeterminate values when the dashed line is between the two rails.

The terms 'edge' and 'clock edge' always refer to the rising edge of the clock. The terms 'asserted' and 'negated' always refer to the globally visible state of the signal on the clock edge, and not to signal transitions. ' ' represents a turnaround cycle in the timing diagrams.

16.4.2.7.1 PCI Read Transactions

This section describes PCI single-beat read transactions and PCI burst read transactions.

A read transaction starts with the address phase, occurring when an initiator asserts PCI*n*_FRAME. During the address phase, PCI*n*_AD contains a valid address and PCI*n*_C/BE contains a valid bus command.

The first data phase of a read transaction requires a turnaround cycle. This allows the transition from the initiator driving PCI*n*_AD as address signals to the target driving PCI*n*_AD as data signals. The turnaround cycle is enforced by the target with the TRDY signal. The target provides valid data at the earliest one cycle after the turnaround cycle. The target must drive the PCI*n*_AD signals when PCI*n*_DEVSEL is asserted.

During the data phase, the PCI*n*_C/BE signals indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The PCI*n*_C/BE signals remain actively driven for both reads and writes from the first clock of the data phase through the end of the transaction.

A data phase completes when data is transferred, which occurs when both PCI*n*_IRDY and PCI*n*_TRDY are asserted on the same clock edge. When either PCI*n*_IRDY or PCI*n*_TRDY is negated, a wait cycle is inserted and no data is transferred. The initiator indicates the last data phase by negating PCI*n*_FRAME when PCI*n*_IRDY is asserted. The transaction is considered complete when data is transferred in the last data phase.

Figure 16-49 illustrates a PCI single-beat read transaction.

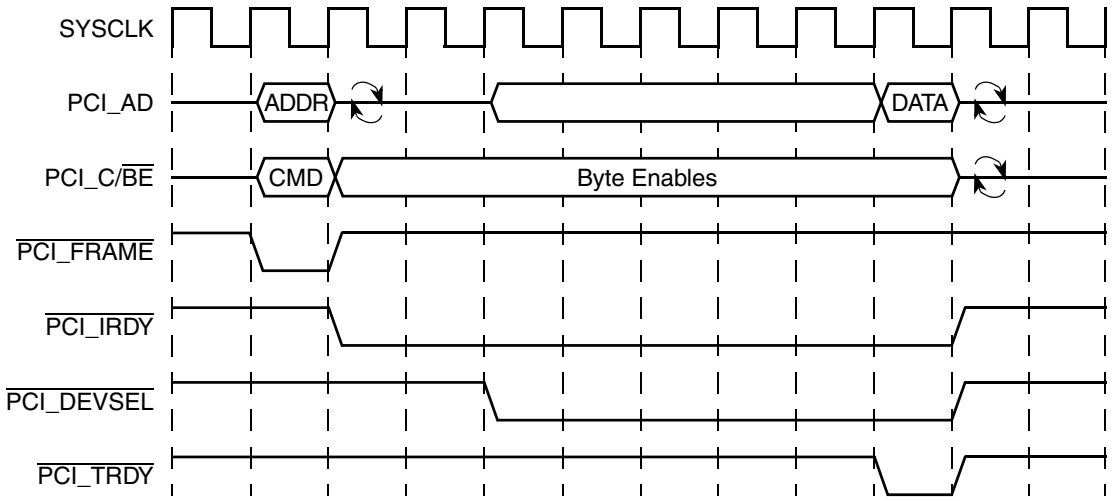


Figure 16-49. PCI Single-Beat Read Transaction

Figure 16-50 illustrates a PCI burst read transaction.

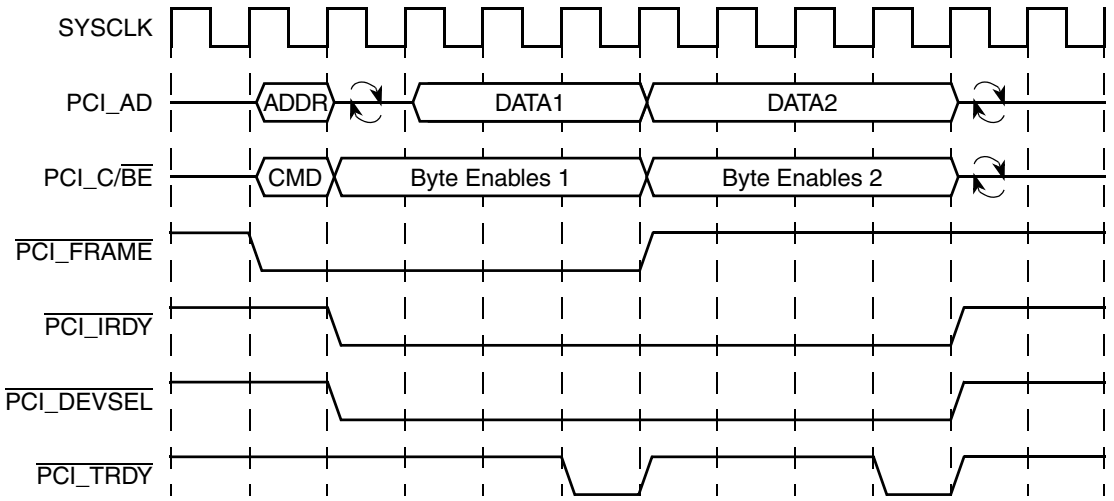


Figure 16-50. PCI Burst Read Transaction

16.4.2.7.2 PCI Write Transactions

This section describes PCI single-beat write transactions, and PCI burst write transactions. A PCI write transaction starts with the address phase, occurring when an initiator asserts PCI*n*_FRAME. 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. The data phases are the same for both read and write transactions. Although not shown in the figures, the initiator must drive the PCI*n*_C/BE signals, even if the initiator is not ready to provide valid data (PCI*n*_IRDY negated).

Figure 16-51 illustrates a PCI single-beat write transaction.

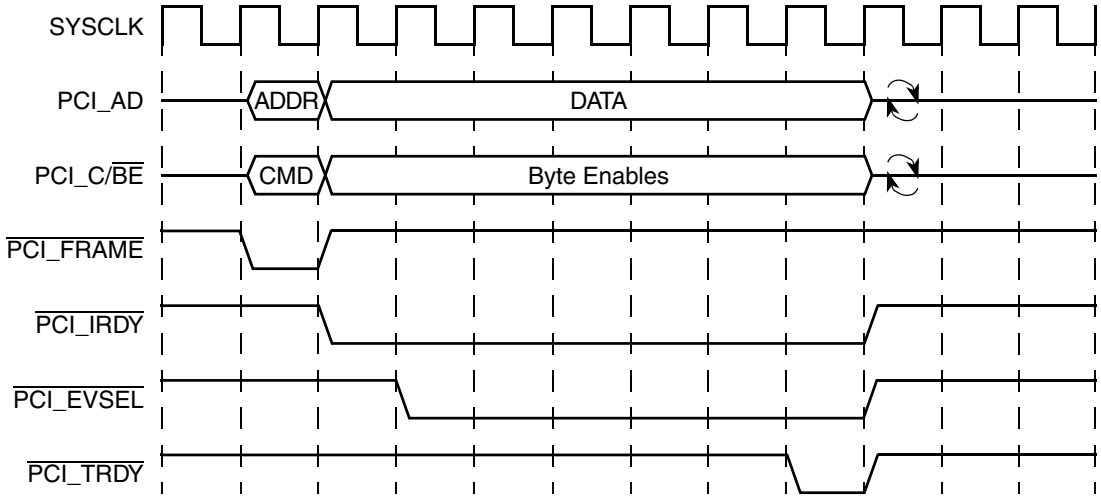


Figure 16-51. PCI Single-Beat Write Transaction

Figure 16-52 illustrates a PCI burst write transaction.

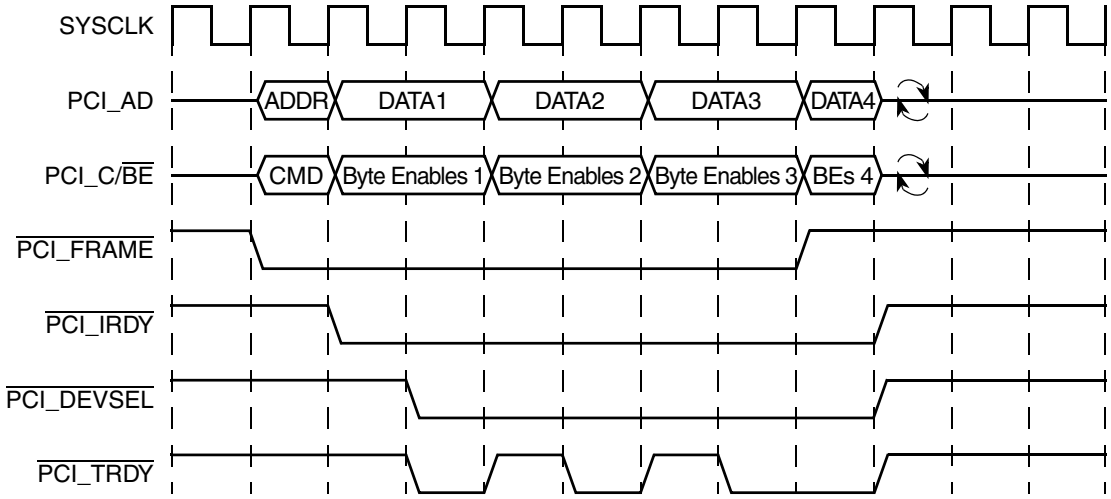


Figure 16-52. PCI Burst Write Transaction

16.4.2.8 Transaction Termination

A PCI transaction may be terminated by either the initiator or the target. The initiator is ultimately responsible for concluding all transactions, regardless of the cause of the termination. All transactions are concluded when PCI*n*_FRAME and PCI*n*_IRDY are both negated, indicating the bus is idle.

16.4.2.8.1 Master-Initiated Termination

Normally, a master initiates termination by negating PCI*n*_FRAME and asserting PCI*n*_IRDY. This indicates to the target that the final data phase is in progress. The final data transfer occurs when both PCI*n*_TRDY and PCI*n*_IRDY are asserted. The transaction is considered complete when data is

transferred in the last data phase. After the final data phase, both PCI*n*_FRAME and PCI*n*_IRDY are negated (the bus becomes idle).

There are three types of master-initiated termination:

- Completion—Refers to termination when the initiator has concluded its intended transaction. This is the most common reason for termination.
- Timeout—Refers to termination when the initiator loses its bus grant (GNT*n* is negated), and its internal latency timer has expired. The intended transaction is not necessarily concluded.
- Master-abort—An abnormal case of master-initiated termination. If no device (including the subtractive decoding agent) asserts PCI*n*_DEVSEL to claim a transaction, the initiator terminates the transaction with a master-abort. For a master-abort termination, the initiator negates PCI*n*_FRAME and then negates PCI*n*_IRDY on the next clock. If a transaction is terminated by master-abort (except for a special-cycle command), the received master-abort bit (bit 13) of the PCI bus status register is set.

As an initiator, if the MPC8555E does not detect the assertion of PCI*n*_DEVSEL within four clock cycles following the address phase (five clock cycles after asserting PCI*n*_FRAME), it terminates the transaction with a master-abort.

16.4.2.8.2 Target-Initiated Termination

By asserting the PCI*n*_STOP signal, a target may request that the initiator terminate the current transaction. Once asserted, the target holds PCI*n*_STOP asserted until the initiator negates PCI*n*_FRAME. Data may or may not be transferred during the request for termination. If PCI*n*_TRDY and PCI*n*_IRDY are asserted during the assertion of PCI*n*_STOP, data is transferred. However, if PCI*n*_TRDY is negated when PCI*n*_STOP is asserted, it indicates that the target will not transfer any more data; therefore, the initiator does not wait for a final data transfer as it would in a completion termination.

When a transaction is terminated by PCI*n*_STOP, the initiator must negate its REQ*n* signal for a minimum of two PCI clock cycles, (one corresponding to when the bus goes to the idle state (PCI*n*_FRAME and PCI*n*_IRDY negated)). If the initiator intends to complete the transaction, it can reassert its REQ*n* immediately following the two clock cycles. If the initiator does not intend to complete the transaction, it can assert REQ*n* whenever it needs to use the PCI bus again.

There are three types of target-initiated termination:

- Disconnect—Disconnect refers to termination requested because the target is temporarily unable to continue bursting. Disconnect implies that some data has been transferred. The initiator may restart the transaction at a later time starting with the address of the next untransferred data. (That is, data transfer may resume where it left off.)
- Retry—Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. Retry implies that no data was transferred. The initiator may start the entire transaction over again at a later time. Note that the *PCI Local Bus Specification*, Rev. 2.2 requires that all retried transactions must be completed.
- Target-Abort—Target-abort is an abnormal case of target-initiated termination. Target-abort is used when a fatal error has occurred or when a target can never respond.

As a target, the MPC8555E terminates a transaction with a target disconnect due to the following:

- It is unable to respond within eight PCI clock cycles (not including the first data phase).
- The transaction is attempting to cross a 4-Kbyte boundary.
- A single beat of data has been transferred and the inbound ATMU is marked non-prefetchable.
- The end of a cache line has been transferred for a cache-wrap mode write transaction. See Section 16.4.2.3.1, "Memory Space Addressing," for more information.

As a target, the MPC8555E responds to a transaction with a retry due to the following:

- The 32-clock latency timer has expired, and the first data phase has not begun.
- There is no more internal buffer space available for an inbound transaction.

Target-abort is indicated by asserting $\overline{\text{PCI}n\text{\_STOP}}$ and negating $\overline{\text{PCI}n\text{\_DEVSEL}}$. This indicates that the target requires termination of the transaction and does not want the transaction retried. If a transaction is terminated by target-abort, the received target-abort bit (bit 12) of the initiator's bus status register and the signaled target-abort bit (bit 11) of the target's bus status register are set. Note that any data transferred in a target-aborted transaction may be corrupt.

For PCI writes to local memory, if an address parity error or data parity error occurs, the MPC8555E aborts the transaction internally, but continues the transaction on the PCI bus.

Figure 16-53 shows several target-initiated terminations.

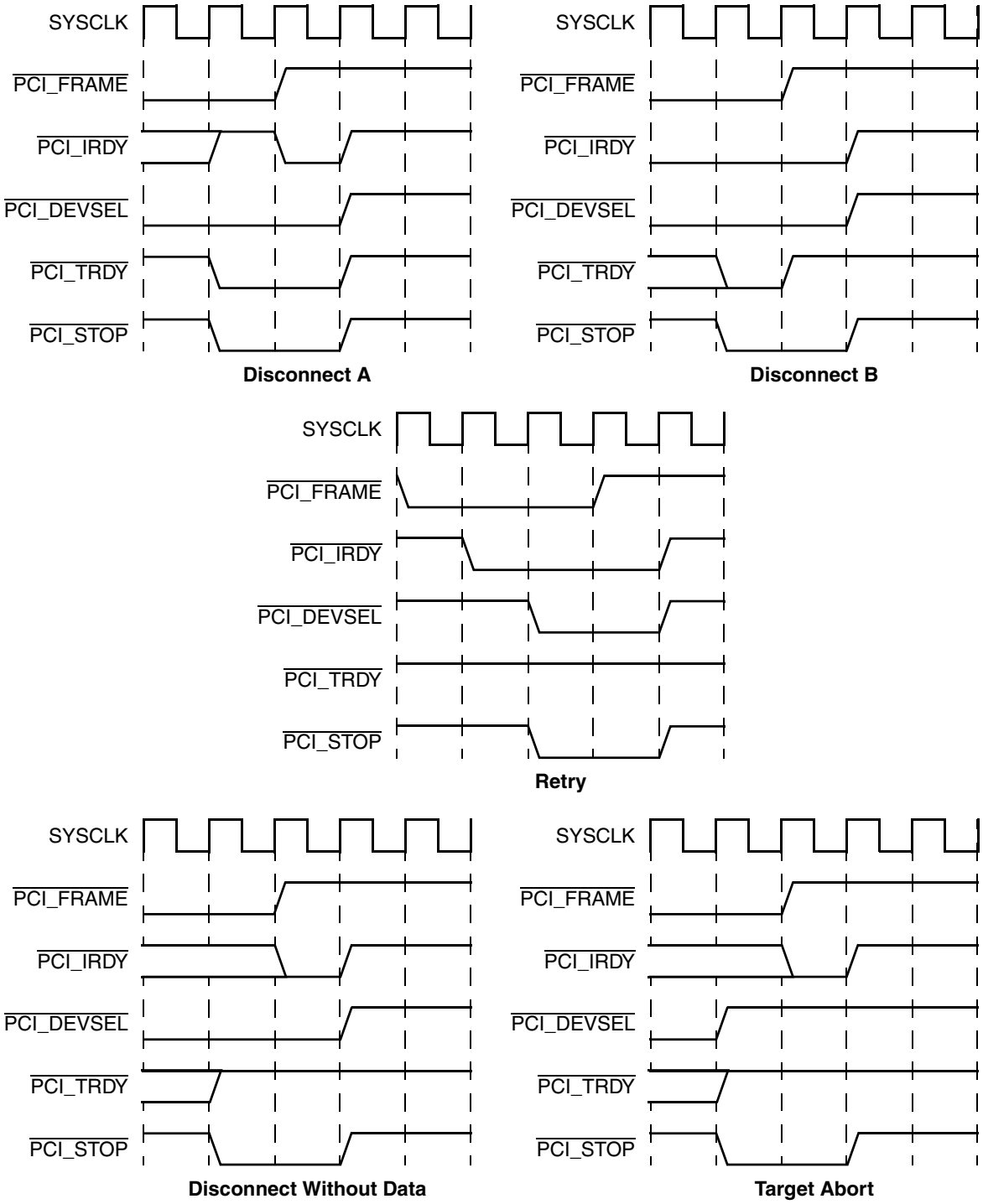


Figure 16-53. PCI Target-Initiated Terminations

The three disconnect terminations are unique in the data transferred at the end of the transaction. For disconnect A, the initiator is negating PCI*n*_IRDY when the target asserts PCI*n*_STOP and data is transferred only at the end of the current data phase. For disconnect B, the target negates PCI*n*_TRDY one

clock after it asserts PCI*n*_STOP, indicating that the target can accept the current data, but no more data can be transferred. For disconnect-without-data, the target asserts PCI*n*_STOP when PCI*n*_TRDY is negated indicating that the target cannot accept any more data.

16.4.2.9 Fast Back-to-Back Transactions

The PCI bus allows fast back-to-back transactions by the same master. During a fast back-to-back transaction, the initiator starts the next transaction immediately without an idle state. The last data phase completes when PCI*n*_FRAME is negated, and PCI*n*_IRDY and PCI*n*_TRDY are asserted. The current master starts another transaction in the clock cycle immediately following the last data transfer for the previous transaction.

Fast back-to-back transactions must avoid contention on the PCI*n*_TRDY, PCI*n*_DEVSEL, PCI*n*_PERR, and PCI*n*_STOP signals. There are two types of fast back-to-back transactions—those that access the same target and those that access multiple targets sequentially. The first type places the burden of avoiding contention on the initiator; the second type places the burden of avoiding contention on all potential targets.

As an initiator, the MPC8555E does not perform any fast back-to-back transactions. As a target, the MPC8555E supports both types of fast back-to-back transactions.

During fast back-to-back transactions, the MPC8555E monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the MPC8555E and the current transaction is directed at the MPC8555E, it delays the assertion of PCI*n*_DEVSEL (as well as PCI*n*_TRDY, PCI*n*_STOP, and PCI*n*_PERR) for one clock cycle to allow the other target to stop driving the bus.

16.4.2.10 Dual Address Cycles

The MPC8555E supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as both an initiator and a target. 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 MPC8555E block supports single-beat and burst DAC transactions.

For the case of the local processor, DAC generation depends on the setting of the POTEARx. If the POTEARx are programmed with nonzero values and a transaction from the local processor core hits in one of the outbound windows, a DAC transaction is generated on the PCI bus with the translated lower 32-bit addresses. Refer to Section 16.3.1.2, “PCI ATMU Outbound Registers,” for more information.

The timing sequence of the PCI signals for single-beat DAC reads is shown in Figure 16-54. The timing for a DAC burst read is shown in Figure 16-55. Figure 16-56 and Figure 16-57 show timing examples for single-beat DAC writes and burst DAC writes, respectively.

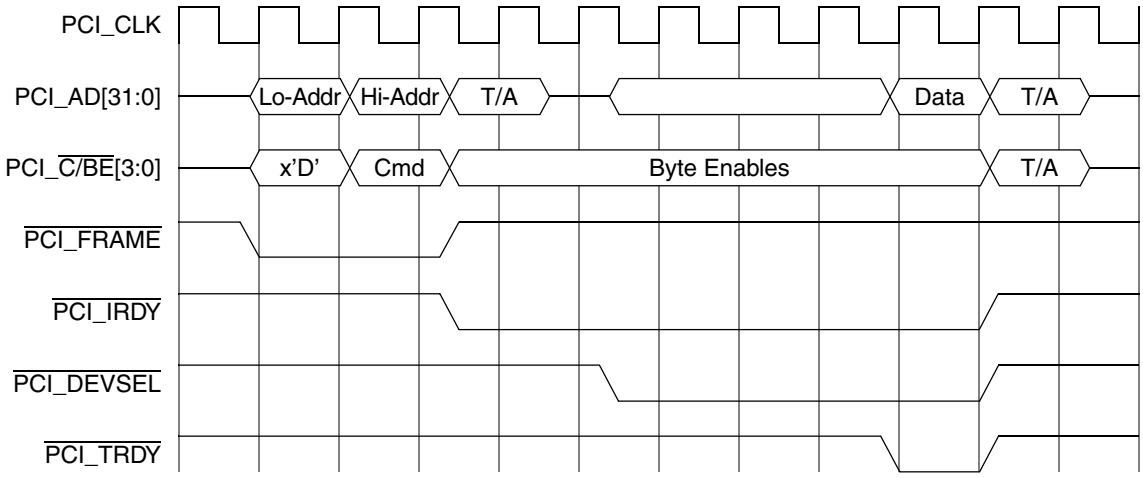


Figure 16-54. DAC Single-Beat Read Example

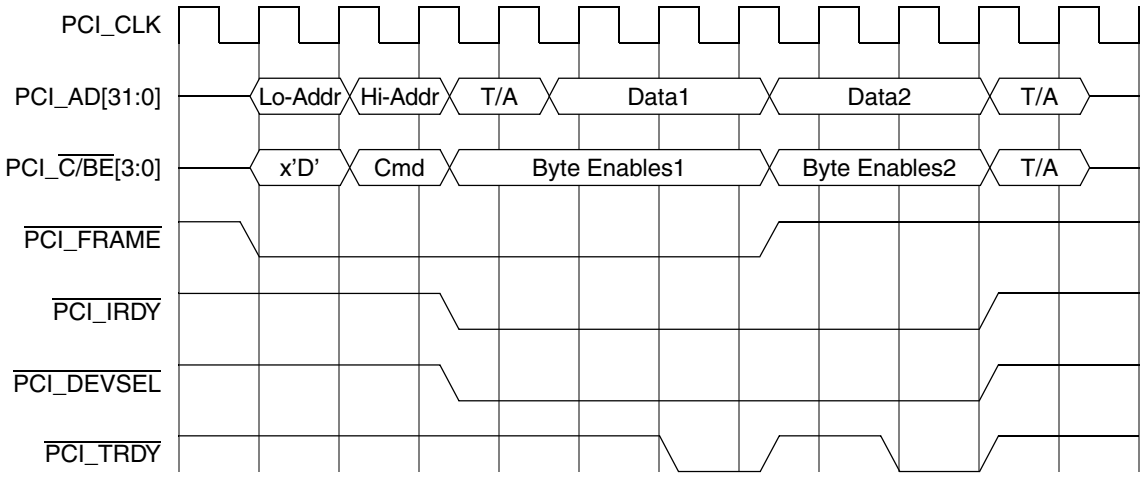


Figure 16-55. DAC Burst Read Example

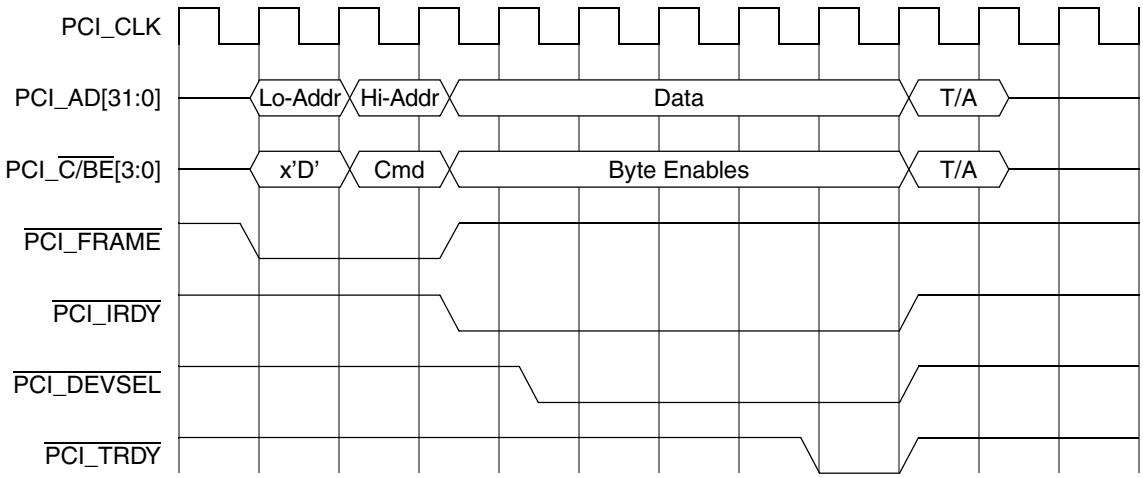


Figure 16-56. DAC Single-Beat Write Example

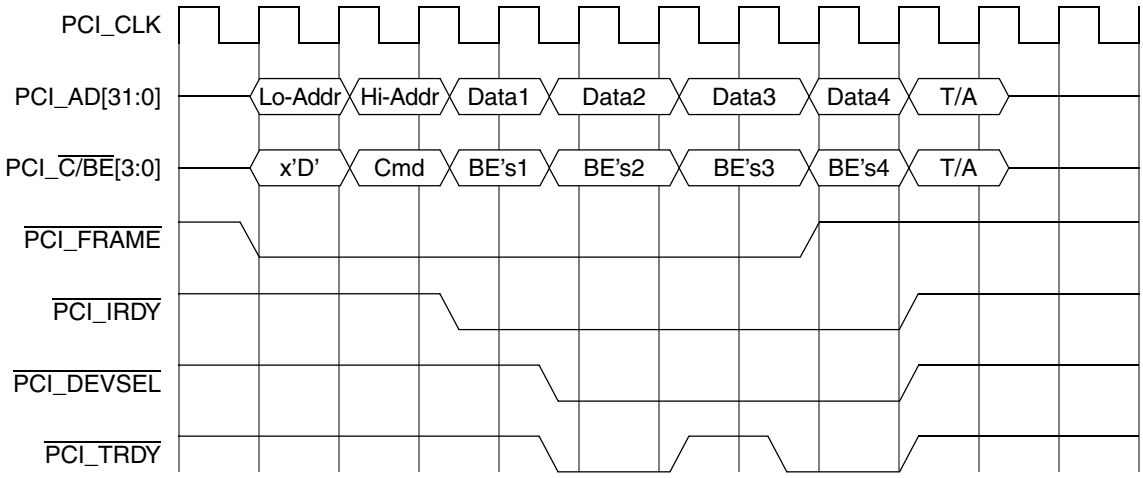


Figure 16-57. DAC Burst Write Example

16.4.2.11 Configuration Cycles

This section describes PCI configuration cycles used for configuring standard PCI devices. The PCI configuration space of any device is intended for configuration, initialization, and catastrophic error-handling functions only. Access to the PCI configuration space should be limited to initialization and error-handling software.

16.4.2.11.1 PCI Configuration Space Header

The first 64 bytes of the 256-byte configuration space consists of a predefined header that every PCI device must support. The predefined header for all PCI devices is shown in Figure 16-58. The first 16 bytes of the predefined header are defined the same for all PCI devices; the remaining 48 bytes of the header may have differing layouts depending on the function of the device. Most PCI devices use the configuration header layout shown in Figure 16-58. The rest of the 256-byte configuration space is device-specific. The PCI header specific to the MPC8555E is described in Section 16.3.2, "PCI Configuration Header."

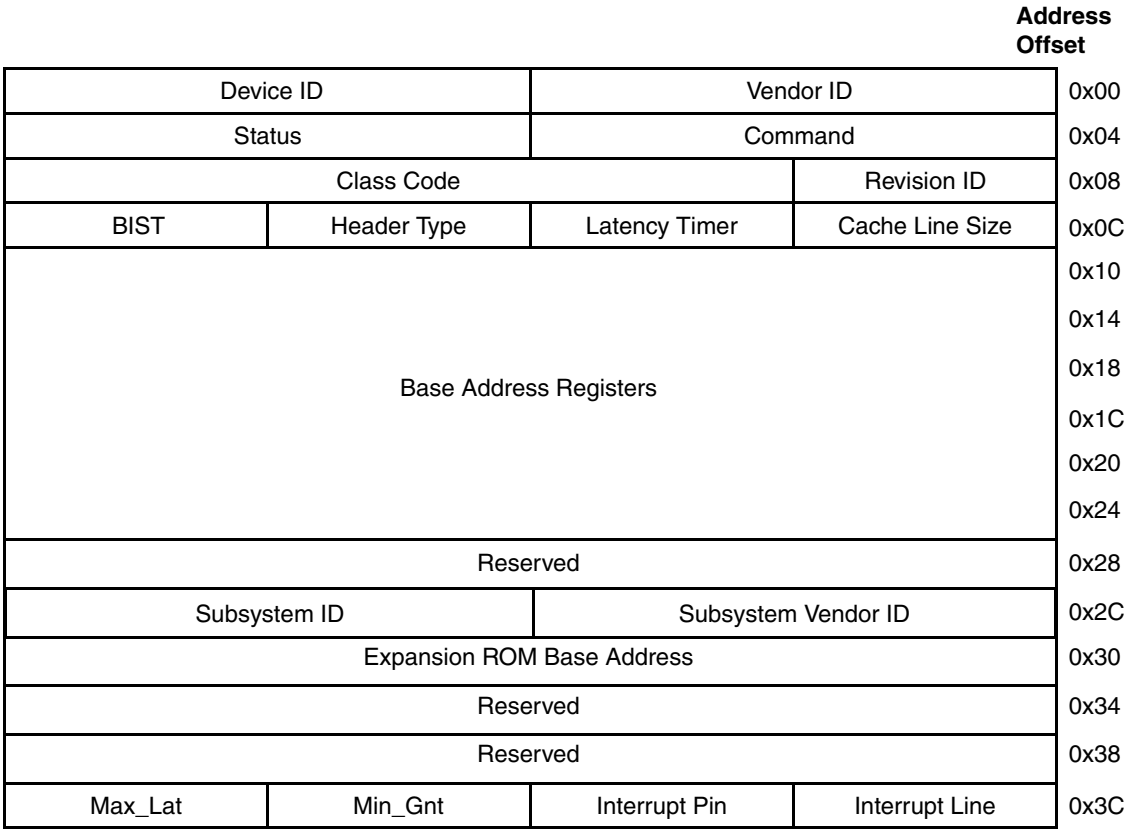


Figure 16-58. Standard PCI Configuration Header

Table 16-49 summarizes the configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*, Rev. 2.2.

Table 16-49. PCI Configuration Space Header Summary

Address Offset (Hex)	Register Name	Description
0x00	Vendor ID	Identifies the manufacturer of the device (assigned by the PCI SIG (special-interest group) to ensure uniqueness).
0x02	Device ID	Identifies the particular device (assigned by the vendor).
0x04	Command	Provides coarse control over a device's ability to generate and respond to PCI bus cycles
0x06	Status	Records status information for PCI bus-related events
0x08	Revision ID	Specifies a device-specific revision code (assigned by vendor)
0x09	Class code	Identifies the generic function of the device and (in some cases) a specific register-level programming interface
0x0C	Cache line size	Specifies the system cache line size in 32-bit units
0x0D	Latency timer	Specifies the value of the latency timer in PCI bus clock units for the device when acting as an initiator
0x0E	Header type	Bits 0–6 identify the layout of bytes 0x10–0x3F; bit 7 indicates a multifunction device. The most common header type (0x00) is shown in Figure 16-58 and in this table.

Table 16-49. PCI Configuration Space Header Summary (continued)

Address Offset (Hex)	Register Name	Description
0x0F	BIST	Optional register for control and status of built-in self test (BIST)
0x10–0x27	Base address registers	Address mapping information for memory and I/O space
0x28	—	Reserved for future use
0x2C	Subsystem Vendor ID	Identifies the subsystem vendor ID (read-only for MPC8555E)
0x2E	Subsystem ID	Identifies the subsystem ID (read-only for MPC8555E)
0x30	Expansion ROM base address	Base address and size information for expansion ROM contained in an add-on board
0x34, 0x38	—	Reserved for future use
0x3C	Interrupt line	Contains interrupt line routing information
0x3D	Interrupt pin	Indicates which interrupt pin the device (or function) uses
0x3E	Min_Gnt	Specifies the length of the device's burst period in 0.25 µs units
0x3F	Max_Lat	Specifies how often the device needs access to the bus in 0.25 µs units

16.4.2.11.2 Accessing the PCI Configuration Space in Host Mode

To access the configuration space, a 32-bit value must be written to the PCI CFG_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. Note that the Bus Master bit in the MPC8555E PCI bus command register must be set before an outbound configuration access is attempted. Device 0 on PCI bus 0 is the MPC8555E itself; thus, device 0, bus 0 is used to access the internal PCI configuration header.

When the MPC8555E detects an access to PCI CFG_DATA, it checks the enable flag and the device number in the PCI CFG_ADDR register. If the enable bit is set, and the device number is not 0b1_1111, the MPC8555E performs a configuration cycle translation function and runs a configuration-read or configuration-write transaction on the PCI bus. If the bus number corresponds to the local PCI bus (bus number = 0x00), the MPC8555E performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the MPC8555E performs a type 1 configuration cycle translation. The device number 0b1_1111 is used for performing interrupt-acknowledge and special-cycle transactions. See Section 16.4.2.12, "Other Bus Transactions," for more information.

See Section 16.3.1.1.1, "PCI Configuration Address Register (CFG_ADDR)," for details on PCI CFG_ADDR and Section 16.3.1.1.2, "PCI Configuration Data Register (CFG_DATA)," for details on PCI CFG_DATA.

Note that because all PCI registers are intrinsically little-endian, in the following examples, the data in the configuration register is shown in little-endian order. Core accesses to the PCI CFG_DATA register should use the load/store with byte-reversed instructions. External PCI masters that use the local address map to access configuration space do not need to reverse bytes since byte lane redirection from the little-endian PCI bus is performed internally.

Example: Configuration sequence, 4-byte data read from the revision ID/standard programming interface/subclass code/class code registers at address offset 0x08 of the PCI configuration header (device 0 on the PCI bus 0 is the MPC8555E itself).

```
Initial values:
         r0 contains 0x8000_0008
         r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR register)
         r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA register)
         r3 contains 0xFFFF_FFFF
         Register at 0x08 contains 0x0B20_0002 (0x0B to 0x08)
Code sequence:
         stw r0, 0 (r1)
         lwbrx r3, 0 (r2)
Results:
         Address CCSRBAR + 0x0_8000 contains 0x8000_0008
         Register r3 contains 0x0B20_0002
```

Example: Configuration sequence, 4-byte data write to PCI register at address offset 0x14 of Device 1 on PCI bus 0.

```
Initial values:
         r0 contains 0x8000_0814
         r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR register)
         r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA register)
         r3 contains 0x1122_3344
         Register at 0x14 contains 0xFFFF_FFFF (0x17 to 0x14)
Code sequence:
         stw r0, 0 (r1)    // Update PCI CFG_ADDR register to point to
                           //register offset 0x14 of device 1.
         stwbrx r3, 0 (r2)
Results:
         Address CCSRBAR + 0x0_8000 contains 0x8000_0814
         Register at 0x14 contains 0x1122_3344 (0x17 to 0x14)
```

Example: Configuration sequence, 2-byte data write to PCI register at address offset 0x1C of Device 1 on PCI bus 0.

```
Initial values:
         r0 contains 0x8000_081C
         r1 contains CCSRBAR + 0x0_8000
         r2 contains CCSRBAR + 0x0_8004
         r3 contains 0xDDCC_BBAA
         Register at 0x1C contains 0xFFFF_FFFF (0x1F to 0x1C)
Code sequence:
         stw r0, 0 (r1)
         sthbrx r3, 0 (r2)
Results:
         Address CCSRBAR + 0x0_8000 contains 0x8000_081C
         Register at 0x1C contains 0xFFFF_BBAA (0x1F to 0x1C)
```

16.4.2.11.3 PCI Configuration in Agent and Agent Lock Modes

In general, agents should not access the configuration space of other external PCI devices. Configuration of agents is a function usually reserved for the host. When the MPC8555E is in agent mode, it responds to remote host-generated PCI configuration cycles. This occurs when a configuration command is decoded along with the IDSEL input signal being asserted. When the MPC8555E is in agent lock mode, it retries

all externally-generated PCI configuration cycles until the ACL bit in the PCI bus function register (0x44) is cleared. See Section 16.5.1, “Power-On Reset Configuration Modes,” for more information.

In either agent or agent lock mode, access to the internal PCI configuration header by the processor core is handled as described in Section 16.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” using device = 0 and bus = 0 in PCI CFG_ADDR to indicate the internal PCI header.

16.4.2.11.4 PCI Type 0 Configuration Translation

Figure 16-59 shows the PCI type 0 translation function performed on the contents of the PCI CFG_ADDR register to the PCI*n*_AD[31:0] signals on the PCI bus during the address phase of the configuration cycle.

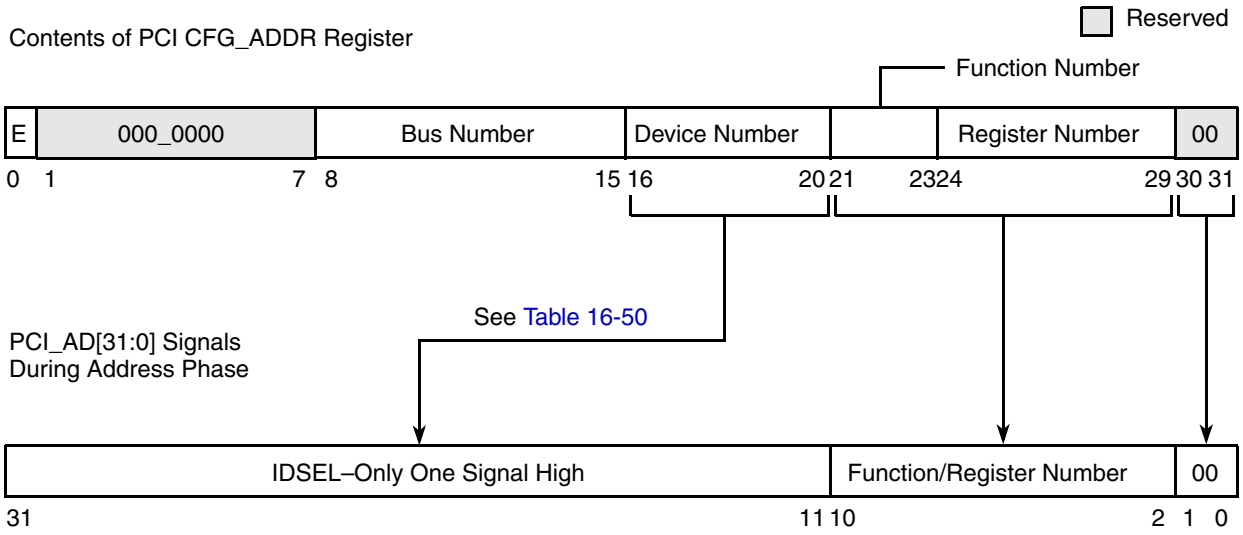


Figure 16-59. PCI Type 0 Configuration Translation

For PCI type 0 configuration cycles, the MPC8555E translates the device number field of the PCI CFG_ADDR register into a unique IDSEL signal for up to 21 different devices. Each device connects its IDSEL input to one of the PCI*n*_AD[31:11] signals. For PCI type 0 configuration cycles, the MPC8555E translates the device number to AD*n* as shown in Table 16-50.

Table 16-50. PCI Type 0 Configuration—Device Number to AD*n* Translation

Device Number		AD*n* Used for IDSEL
Binary	Decimal	
0b0_0000	0	—[1]
0b0_0001–0b0_1001	1–9	—[2]
0b0_1010	10	AD31
0b0_1011	11	AD11
0b0_1100	12	AD12
0b0_1101	13	AD13
0b0_1110	14	AD14

Table 16-50. PCI Type 0 Configuration—Device Number to AD*n* Translation (continued)

Device Number		AD*n* Used for IDSEL
Binary	Decimal	
0b0_1111	15	AD15
0b1_0000	16	AD16
0b1_0001	17	AD17
0b1_0010	18	AD18
0b1_0011	19	AD19
0b1_0100	20	AD20
0b1_0101	21	AD21
0b1_0110	22	AD22
0b1_0111	23	AD23
0b1_1000	24	AD24
0b1_1001	25	AD25
0b1_1010	26	AD26
0b1_1011	27	AD27
0b1_1100	28	AD28
0b1_1101	29	AD29
0b1_1110	30	AD30
0b1_1111 [3]	31	—

[1] No external configuration transaction takes place; rather, internal registers are accessed.

[2] No IDSEL line asserted. Type0 configuration transaction is run, but ends with a master abort since no device responds.

[3] A device number of all ones indicates a PCI special-cycle or interrupt-acknowledge transaction.

For PCI type 0 translations, the function number and register number fields are copied without modification onto the PCI*n*_AD[10:2] signals during the address phase. The PCI*n*_AD[1:0] signals are driven to 0b00 during the address phase for type 0 configuration cycles. The MPC8555E implements address stepping on configuration cycles so that the target's IDSEL, which is connected directly to one of the PCI_AD lines, reaches a stable value. This means that a valid address and command are driven on PCI*n*_AD[31:0] and PCI*n*_C/$\overline{\text{BE}}$[3:0] one clock cycle before the assertion of $\overline{\text{PCI}n\text{\_FRAME}}$.

16.4.2.11.5 PCI Type 1 Configuration Translation

For type 1 translations, the MPC8555E copies the 30 high-order bits of the PCI CFG_ADDR register (without modification) onto the PCI*n*_AD[31:2] signals during the address phase. The MPC8555E

automatically translates PCI*n*_AD[1:0] into 0b01 during the address phase to indicate a type 1 configuration cycle.

16.4.2.12 Other Bus Transactions

There are two other PCI transactions that the MPC8555E supports—interrupt acknowledge and special cycles. As an initiator, the MPC8555E may initiate both interrupt acknowledge and special-cycle transactions; however, as a target, the MPC8555E ignores interrupt-acknowledge and special-cycle transactions. Both transactions make use of the PCI CFG_ADDR and PCI CFG_DATA registers described in Section 16.4.2.11.3, "PCI Configuration in Agent and Agent Lock Modes."

16.4.2.12.1 Interrupt-Acknowledge Transactions

The PCI bus supports an interrupt-acknowledge transaction. The interrupt-acknowledge command is a read operation implicitly addressed to the system interrupt controller. Note that the PCI interrupt-acknowledge command does not address the MPC8555E PIC processor interrupt-acknowledge register and does not return the interrupt vector address from the PIC unit. See Chapter 10, "Programmable Interrupt Controller," for more information about the PIC unit.

When the MPC8555E detects a read to the PCI CFG_DATA register, it checks the enable flag and the device number in the PCI CFG_ADDR register. If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all ones (0b1_1111), the function number is all ones (0b111), and the register number is zero (0b00_0000), then the MPC8555E performs an interrupt-acknowledge transaction. If the bus number indicates a nonlocal PCI bus, the MPC8555E performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

The address phase contains no valid information other than the interrupt-acknowledge command (PCI*n*_C/$\overline{\text{BE}}$[3:0] = 0b0000). Although there is no explicit address, PCI_AD are driven to a stable state, and parity is generated. Only one device (the system interrupt controller) on the PCI bus should respond to the interrupt-acknowledge command by asserting $\overline{\text{PCI}n\text{\_DEVSEL}}$. All other devices on the bus should ignore the interrupt-acknowledge command. As stated previously, the MPC8555E PIC unit does not respond to PCI interrupt-acknowledge commands.

During the data phase, the responding device returns the interrupt vector on PCI_AD when $\overline{\text{PCI}n\text{\_TRDY}}$ is asserted. The size of the interrupt vector returned is indicated by the value driven on the PCI*n*_C/$\overline{\text{BE}}$ signals.

The MPC8555E also provides a direct way to generate PCI interrupt-acknowledge transactions. Reads from PCI INT_ACK at offset 0x0_8008 generate PCI interrupt-acknowledge transactions. Note that processor writes to these addresses do nothing.

16.4.2.12.2 Special-Cycle Transactions

The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. The special-cycle command contains no explicit destination address but is broadcast to all PCI agents.

When the MPC8555E detects a write to PCI CFG_DATA, it checks the enable flag and the device number in PCI CFG_ADDR. If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all ones (0b1_1111), the function number is all ones (0b111), and the register number is zero (0b00_0000), then the MPC8555E performs a special-cycle transaction on the local PCI bus. If the bus number indicates a nonlocal PCI bus, the MPC8555E performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

Aside from the special-cycle command (PCI*n*_C/$\overline{\text{BE}}$[3:0] = 0b0001) the address phase contains no other valid information. Although there is no explicit address, PCI_AD are driven to a stable state, and parity is generated. During the data phase, PCI*n*_AD contain the special-cycle message and an optional data field. The special-cycle message is encoded on the 16 least-significant bits (PCI*n*_AD[15:0]); the optional data field is encoded on the most-significant 16 lines (PCI*n*_AD[31:16]). The special-cycle message encodings are assigned by the PCI SIG steering committee. The current list of defined encodings are provided in Table 16-51.

Table 16-51. Special-Cycle Message Encodings

PCI*n*_AD[15:0]	Message
0x0000	SHUTDOWN
0x0001	HALT
0x0002	x86 architecture-specific
0x0003–0xFFFF	—

Note that the MPC8555E does not automatically issue a special-cycle message when it enters any of its power-saving modes. It is the responsibility of software to issue the appropriate special-cycle message, if needed.

Each receiving agent must determine whether the special-cycle message is applicable to itself. Assertion of $\overline{\text{PCI}n\text{\_DEVSEL}}$ in response to a special-cycle command is not necessary. The initiator of the special-cycle transaction can insert wait states but since there is no specific target, the special-cycle message and optional data field are valid on the first clock $\overline{\text{PCI}n\text{\_IRDY}}$ is asserted. All special-cycle transactions are terminated by master-abort; however, the master-abort bit in the initiator's bus status register is not set for special-cycle terminations.

16.4.2.13 PCI Error Functions

PCI provides for parity and other system errors to be detected and reported. This section describes generation and detection of parity and error reporting for the PCI bus.

16.4.2.13.1 PCI Parity

Generating parity is not optional; it must be performed by all PCI-compliant devices. All PCI transactions, regardless of type, calculate even parity; that is, the number of ones on the PCI*n*_AD[31:0], PCI*n*_C/$\overline{\text{BE}}$[3:0], and PCI*n*_PAR signals all sum to an even number and the number of ones on the PCI1_AD[63:33], PCI1_C/$\overline{\text{BE}}$[7:4], and PCI1_PAR64 signals all sum to an even number.

Parity provides a way to determine, on each transaction, if the initiator successfully addressed the target and transferred valid data. The PCI1_C/$\overline{BE}$[7:4] and PCI*n*_C/$\overline{BE}$[3:0] signals are included in the parity calculation to ensure that the correct bus command is performed (during the address phase) and correct data is transferred (during the data phase). The agent responsible for driving the bus must also drive even parity on the PAR and PCI1_PAR64 signals one clock cycle after a valid address phase or valid data transfer, as shown in Figure 16-60.

During the address and data phases, parity covers all 32 or 64 address/data signals and 4 or 8 command/byte enable signals, regardless of whether all lines carry meaningful information. Byte lanes not actually transferring data must contain stable (albeit meaningless) data and are included in parity calculation. During configuration, special-cycle, or interrupt-acknowledge commands; some address lines are not defined, but are driven to stable values and are included in parity calculation.

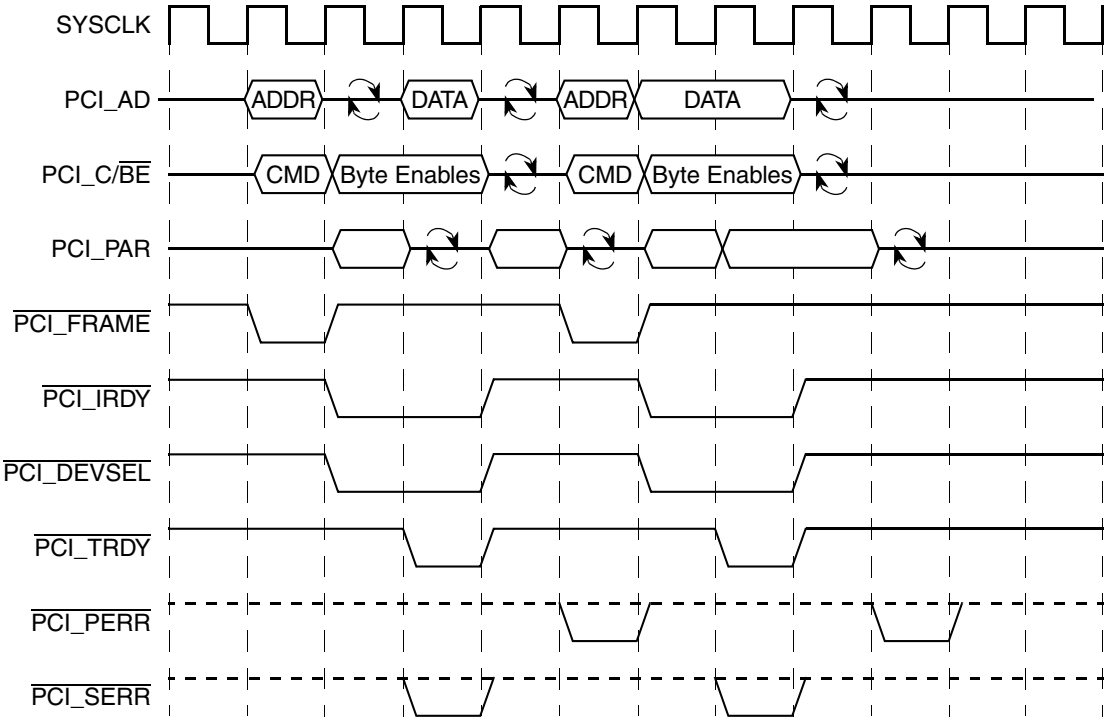


Figure 16-60. PCI Parity Operation

16.4.2.13.2 Error Reporting

PCI provides for the detection and signaling of both parity and other system errors. Two signals are used to report these errors—PCI*n*_PERR and PCI*n*_SERR. The PCI*n*_PERR signal is used exclusively to report data parity errors on all transactions except special cycles. The PCI*n*_SERR signal is used for other error signaling including address parity errors and data parity errors on special-cycle transactions; it may also be used to signal other system errors.

Note that some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI error enable register (ERR_EN). Likewise, some errors are reported in two bits—one in the PCI bus status register and another in the PCI error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr

abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur.

Table 16-52 shows the actions taken for each kind of error.

Table 16-52. PCI Mode Error Actions

PCI Error Type	Error Detect Register bit	PCI Status Register bit	Comment
PCI Outbound Read			
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$	—	No data transferred
Received parity error for data phase	Mstr $\overline{\text{PERR}}$	Detected parity error, master data parity error detected	No data transferred
Master abort	Mstr abort	Received master abort	No data transferred
Target abort	Trgt abort	Received target abort	No data transferred
Memory space violation	ORMSV	—	No data transferred. Only 8 bytes are requested in PCI bus.
PCI Outbound Write			
Received $\overline{\text{SERR}}$ related to address phase	Rcvd $\overline{\text{SERR}}$	—	May float AD bus to avoid contention
Received $\overline{\text{SERR}}$ related to data phase	Rcvd $\overline{\text{SERR}}$	—	
Received $\overline{\text{PERR}}$ (data phase)	Mstr $\overline{\text{PERR}}$	Master data parity error	
Master abort	Mstr abort	Received master abort	
Target abort	Trgt abort	Received target abort	
Memory space violation	OWMSV	—	Only 8 bytes transferred.
PCI Inbound Read			
Detected parity error for address phase	Addr parity error	Detected parity error, signaled system error	Float AD bus
Detected parity error on upper address bus for address phase (SAC or DAC)	—	—	No PAR64 check during address phase
Received $\overline{\text{SERR}}$ at any phase	Received $\overline{\text{SERR}}$	—	
Received $\overline{\text{PERR}}$ (data phase)	Target $\overline{\text{PERR}}$	—	
Internal error	Target abort	Signaled target abort	

Table 16-52. PCI Mode Error Actions (continued)

PCI Error Type	Error Detect Register bit	PCI Status Register bit	Comment
PCI Inbound Write			
Detected parity error for address phase	Addr parity error	Detected parity error, signaled system error	Cache line purged
Detected parity error on upper address bus for Address phase (SAC or DAC)	—	—	No PAR64 check during address phase
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$	—	
Detected parity error for data phase	Trgt $\overline{\text{PERR}}$	Detected parity error	Cache line purged

16.5 Initialization/Application Information

This section describes some tips for use of the PCI controllers.

16.5.1 Power-On Reset Configuration Modes

The PCI1 interface can power-on in three modes: host mode, agent mode and agent configuration lock mode. Certain bits in the configuration registers are set differently according to the POR (power-on reset) mode. Also, certain configuration bits have different implications when compared with past Freescale parts and PCI implementations. Note that after reset, the device cannot be switched from one mode to another. Also note that the PCI2 interface only supports host mode.

The affected configuration bits are defined in Table 16-53

Table 16-53. Affected Configuration Register Bits for POR

Register (offset)	Bit	Name	Register Description
PCI command register (0x04)	2	Bus master	Controls whether the device can master a transaction on the PCI bus. If cleared, the device can not master a transaction. This bit is independent of host or agent mode.
	1	Memory space	Controls the acknowledgement of inbound memory transactions. If cleared, all inbound memory accesses (including accesses to PCSRBAR space) end in a master abort. This bit is independent of host or agent mode.
PCI bus function register (0x44)	5	ACL	Valid only in agent mode. Controls acknowledgement of inbound configuration accesses. If set, all inbound configuration accesses are retried. If cleared, inbound configuration accesses are acknowledged. In host mode all inbound configuration accesses end in master aborts.
	0	PAH	Determines whether the device is in agent or host mode. Zero indicates host mode.

The POR reset values for the affected configuration bits are described in Table 16-54.

Table 16-54. Power-On Reset Values for Affected Configuration Bits

Mode	Configuration Bit			
	Bus Master	Memory Space	ACL	PAH
Host	PCI1: 1, PCI2: 1	PCI1: 0, PCI2: 0	PCI1: X, PCI2: 0	PCI1: 0, PCI2: 0
Agent	PCI1: 0, PCI2: 0	PCI1: 0, PCI2: 0	PCI1: 0, PCI2: 0	PCI1: 1, PCI2: 0
Agent configuration lock	PCI1: 0, PCI2: 0	PCI1: 0, PCI2: 0	PCI1: 1, PCI2: 0	PCI1: 1, PCI2: 0

16.5.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). The ACL bit is a do not care. The device powers up with the ability to master transactions on the PCI bus, however in order to acknowledge memory transactions, the memory space bit must be set.

16.5.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. However the device cannot master transactions or acknowledge inbound memory accesses on the PCI bus until the appropriate configuration bits (bus master and memory space, respectively) have been set.

16.5.1.3 Agent Configuration Lock Mode

Agent configuration lock mode is similar to agent mode with the added restriction that when the device powers up in agent configuration lock mode, it retries all inbound configuration accesses until the ACL bit is cleared. The purpose of this mode is to allow initial configuration on the port by the local processor before opening the port to be further configured by the external host. As in agent mode, the device in agent configuration lock mode cannot master transactions or acknowledge inbound memory accesses on the PCI bus until the appropriate configuration bits (bus master and memory space, respectively) have been set.

16.5.2 Extended 64-Bit PCI1 Signal Connections

The device can be configured as one PCI-64 interface or two independent PCI-32 interfaces at power-on reset. See Section 16.1.3.2, “PCI-64 or Two PCI-32 Interface Configuration for more details.

When this device is configured as one PCI-64 interface, PCI2 pins are used for the extended 64-bit PCI1 signals. Table 16-55 shows the extended 64-bit PCI1 signal connections using the PCI2 pins.

Table 16-55. Extended 64-Bit PCI1 Signal Connections

PCI1 PIN Names	PCI2 PIN Names
PCI1_AD[63:32]	PCI2_AD[31:0]
PCI1_C/$\overline{\text{BE}}$[7:4]	PCI2_C/$\overline{\text{BE}}$[3:0]
$\overline{\text{PCI1\_REQ64}}$	$\overline{\text{PCI2\_FRAME}}$
$\overline{\text{PCI1\_ACK64}}$	$\overline{\text{PCI2\_DEVSEL}}$
PCI1_PAR64	PCI2_PAR

Chapter 17
Security Engine (SEC) 2.0

This chapter describes the functionality of the MPC8555E integrated security engine (SEC 2.0). It addresses the following topics:

The SEC 2.0 is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption, from the processor core of the MPC8555E. It is optimized to process all algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE 802.11i standard. The SEC 2.0 is derived from integrated security cores found in other members of the PowerQUICC family, including SEC 1.0, the version implemented in the MPC8272.

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
    - Programmable field size up to 2048 bits
  - Elliptic curve cryptography
    - F_2m and F(p) modes
    - Programmable field size up to 511 bits
- DEU—Data Encryption Standard execution unit
  - DES, 3DES
  - Two key (K1, K2, K1) or three key (K1, K2, K3)
  - ECB and CBC modes for both DES and 3DES
- AESU—Advanced encryption standard execution 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- or 256-bit message digest
  - MD5 with 128-bit message digest
  - HMAC with either algorithm
- RNG—Random number generator
- Master/slave logic, with DMA
  - 32-bit address/64-bit data
  - Up to 166-MHz operation
  - 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 2.0 to concatenate multiple segments of memory when reading input data
  - Similarly, scatter capability enables SEC 2.0 to write to multiple segments of memory when writing output data

17.1 Architecture Overview

The SEC 2.0 (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 17-1 shows that the SEC communicates with other modules through the internal bus.

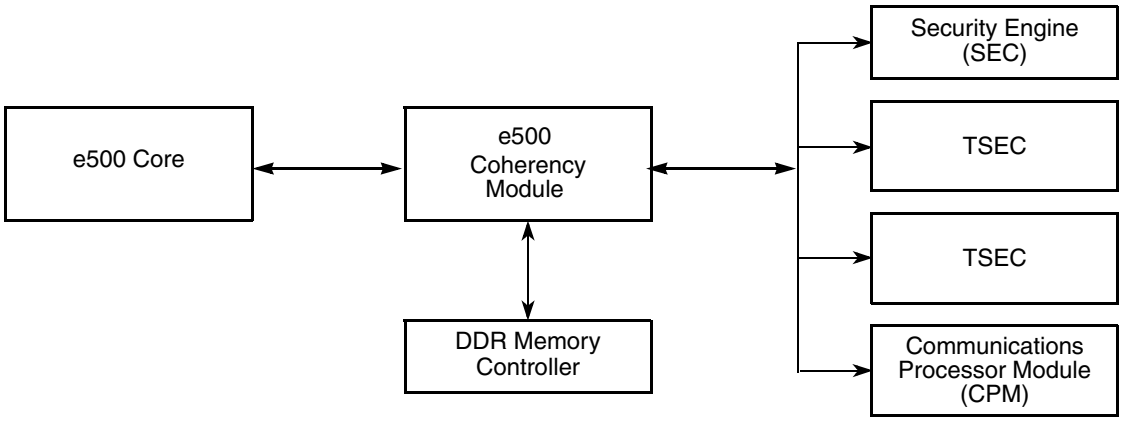


Figure 17-1. SEC Connected to MPC8555E Internal Bus

A block diagram of the SEC internal architecture is shown in Figure 17-2. The bus interface module is designed to transfer 64-bit words between the bus and any register inside the SEC.

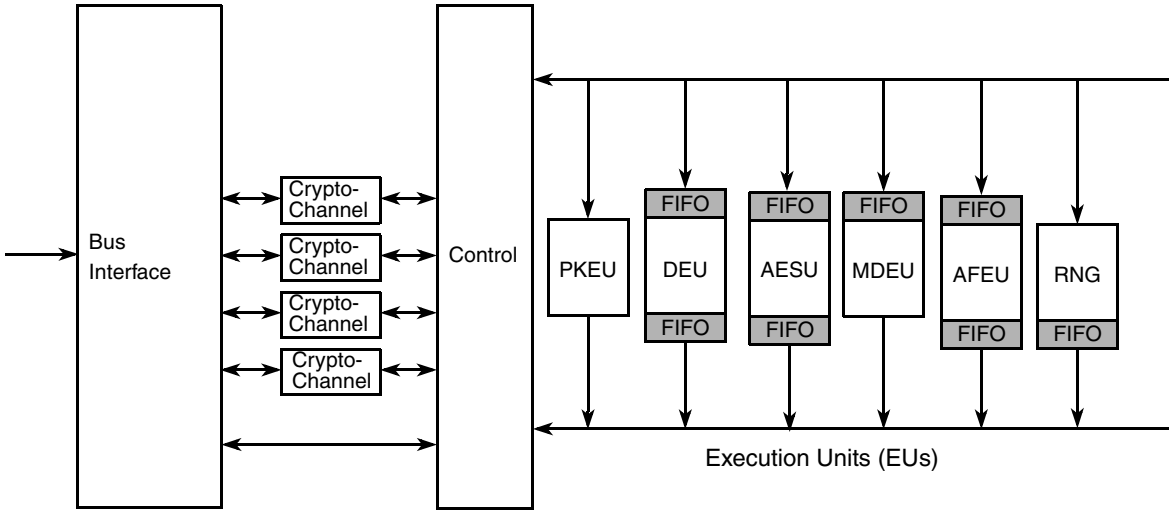


Figure 17-2. SEC Functional Modules

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 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, initialization vectors (IVs), and data from locations specified in the rest of the descriptor. The controller satisfies the requests by making requests to the master interface per the programmable priority scheme. Data is fed into the execution units through their registers and input FIFOs. The execution units read from their input FIFOs and write processed data to their output FIFOs. The channel requests 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.

17.1.1 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 17-1.

Table 17-1. 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.

Upon completion of the current descriptor, the channel checks the next entry in its fetch FIFO, and, if it is nonzero, the channel is instructed to request a burst read of the next descriptor.

Processing of the next descriptor (and whether or not a done signal is generated) is determined by the programming of the crypto-channel's configuration register. Two modes of operation are supported:

- Signal done at end of every descriptor
- Signal done at end of a selected descriptor

The crypto-channel can signal done through an interrupt or by a write-back of the descriptor header after processing a data packet descriptor. The value written back is identical to that of the header, with the exception that a done field is set.

Occasionally, a descriptor field may not be applicable to the requested service. For example, if using DES in ECB mode, the contents of the IV field do not affect the result of the DES computation. Therefore, when processing data packet descriptors, the crypto-channel skips any pointer that has an associated length of zero.

For more information, refer to Section 17.1.1, "Data Packet Descriptors."

17.1.2 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 standard 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 RC4 compatible stream cipher symmetric-key cryptography
- AESU—For performing the advanced encryption standard algorithm
- 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 17.4, "Execution Units."

17.1.2.1 Public Key Execution Unit (PKEU)

The PKEU is capable of performing many advanced mathematical functions to support both RSA and ECC public key cryptographic algorithms. ECC is supported in both F_2m (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 17.4.1, "Public Key Execution Unit (PKEU)."

17.1.2.1.1 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 increments of 8, with each value *i* supporting all actual field sizes from 8*i* – 7 to 8*i*. 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.

17.1.2.1.2 Modular Exponentiation Operations

The PKEU is also capable of performing 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 mod N
- A′ E mod N
- (A × B) R^{-1} mod N
- (A × B) R^{-2} mod N
- (A + B) mod N
- (A – B) mod N

Where the following variable definitions: A′ = AR mod 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 105 bits. 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 exponentiation.

17.1.2.2 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 clock chaining (CBC).

For more information, refer to Section 17.4.2, "Data Encryption Standard Execution Unit (DEU)."

17.1.2.3 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 17.4.3, “ARC Four Execution Unit (AFEU).”

17.1.2.4 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.

For more information, refer to Section 17.4.6, “Advanced Encryption Standard Execution Unit (AESU).”

17.1.2.5 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 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-256 is a 256-bit hash function that provides 256 bits of security against collision attacks.
- The MDEU also supports HMAC computations, as specified in RFC 2104.

For more information, refer to Section 17.4.4, “Message Digest Execution Unit (MDEU).”

17.1.2.6 Random Number Generator (RNG)

The RNG is a digital integrated circuit 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 17.4.5, “Random Number Generator (RNG).”

17.1.3 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

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 SEC2.0, 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 generate two types of done notification signals when it completes operation on a descriptor. It can signal done through an interrupt or by a writeback of the descriptor header after processing a descriptor. The value written back is identical to that of the header, with the exception that a done field is set.

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 17.5, “Crypto-Channels.”

17.1.4 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.
- Dynamic EU access—A crypto-channel can request a particular service from any available execution unit.

17.1.4.1 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.

17.1.4.2 Dynamic EU Access

Processing begins when a data packet descriptor pointer is written to the fetch FIFO of one of the crypto-channels. Prior to fetching the data referred to by the descriptor and based on the services requested by the descriptor header in the descriptor buffer, the controller dynamically reserves usage of an EU to the crypto-channel. If all appropriate EUs are already dynamically reserved by other crypto-channels, the crypto-channel stalls and waits to fetch data until an appropriate EU is available.

If multiple crypto-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 crypto-channel fetches and loads the appropriate data packets, operates the EU, unloads data to system memory, and releases the EU for use by another crypto-channel.

For more information, refer to Section 17.6, “SEC Controller.”

17.1.5 Bus Interface

The master/slave interface manages communication between the SEC’s internal execution units and the MPC8555E 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 17.7, "Bus Interface."

17.2 Configuration of Internal Memory Space

Table 17-2 shows the base address map, while Table 17-3 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 CCSRBAR. Refer to Section 2.3, "Configuration, Control, and Status Register 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 17-2. SEC Base Address Map

Offset (AD 17–0)	Module	Description	Type	Reference
0x3_0000–0x3_0FFF	—	Reserved	—	—
0x3_1000–0x3_10FF	Controller	Arbiter/controller control register space	Resource control	17.6/17-92
0x3_1100–0x3_11FF	Channel_1	Crypto-channel 1	Data control	17.5/17-80
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		
0x3_2000–0x3_2FFF	DEU	DES/3DES execution unit	Crypto EU	17.4.2/17-33
0x3_4000–0x3_4FFF	AESU	AES execution unit		17.4.6/17-67
0x3_6000–0x3_6FFF	MDEU	Message digest execution unit		17.4.4/17-51
0x3_8000–0x3_8FFF	AFEU	ARC Four execution unit		17.4.3/17-42
0x3_A000–0x3_AFFF	RNG	Random number generator		17.4.5/17-61
0x3_C000–0x3_CFFF	PKEU	Public key execution unit		17.4.1/17-24

Table 17-3 shows the system address map showing all functional registers. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 17-3. SEC Address Map

Offset	Register	Access	Reset	Section/Page
Controller Registers				
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000_0000_0000	17.6.2.1/17-93
0x3_1010	ISR—Interrupt status register	R	0x0000_0000_0000_0000	17.6.2.2/17-94

Table 17-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000_0000_0000	17.6.2.3/17-95
0x3_1020	ID—Identification register	R	0x0000_0000_0000_0040	17.6.2.4/17-97
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	17.6.2/17-92
0x3_1030	MCR—Master control register	R/W	0000_0000_0000_0000	17.6.2.5/17-98
Channel 1				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	0x0000_0000_0000_0000	17.5.1.1/17-81
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000_0000_0007	17.5.1.2/17-83
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	0x0000_0000_0000_0000	17.5.1.3/17-89
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	0x0000_0000_0000_0000	17.5.1.4/17-90
0x3_1180–0x3_11BF	DBn—Crypto-channel 1descriptor buffers[0-7]	R	0x0000_0000_0000_0000	17.5.1.5/17-90
Channel 2				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	0x0000_0000_0000_0000	17.5.1.1/17-81
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000_0000_0007	17.5.1.2/17-83
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	0x0000_0000_0000_0000	17.5.1.3/17-89
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	0x0000_0000_0000_0000	17.5.1.4/17-90
0x3_1280–0x3_12BF	DBn—Crypto-channel 2 descriptor buffers[0-7]	R	0x0000_0000_0000_0000	17.5.1.5/17-90
Channel 3				
0x3_1308	CCCR3—Crypto-channel 3 configuration register	R/W	0x0000_0000_0000_0000	17.5.1.1/17-81
0x3_1310	CCPSR3—Crypto-channel 3 pointer status register	R	0x0000_0000_0000_0007	17.5.1.2/17-83
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	0x0000_0000_0000_0000	17.5.1.3/17-89
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	0x0000_0000_0000_0000	17.5.1.4/17-90
0x3_1380–0x3_13BF	DBn—Crypto-channel 3 descriptor buffers[0-7]	R	0x0000_0000_0000_0000	17.5.1.5/17-90
Channel 4				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	0x0000_0000_0000_0000	17.5.1.1/17-81
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000_0000_0007	17.5.1.2/17-83
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	0x0000_0000_0000_0000	17.5.1.3/17-89
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	0x0000_0000_0000_0000	17.5.1.4/17-90
0x3_1480–0x3_14BF	DBn—Crypto-channel 4 descriptor buffers[0-7]	R	0x0000_0000_0000_0000	17.5.1.5/17-90
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000_0000_0000	17.4.2.1/17-33

Table 17-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000_0000_0000	17.4.2.2/17-34
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000_0000_0000	17.4.2.3/17-35
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000_0000_0000	17.4.2.4/17-36
0x3_2028	DEUSR—DEU status register	R	0x0000_0000_0000_0000	17.4.2.5/17-37
0x3_2030	DEUISR—DEU interrupt status register	R	0x0000_0000_0000_0000	17.4.2.6/17-38
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000_0000_3000	17.4.2.7/17-39
0x3_2050	DEUEUG—DEU EU-Go register	W	0x0000_0000_0000_0000	17.4.2.8/17-41
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000_0000_0000	17.4.2.9/17-41
0x3_2400	DEUK1—DEU key register 1	W	—	17.4.2.10/17-41
0x3_2408	DEUK2—DEU key register 2	W	—	17.4.2.10/17-41
0x3_2410	DEUK3—DEU key register 3	W	—	17.4.2.10/17-41
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000_0000_0000	17.4.2.11/17-42
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000_0000_0000	17.4.6.1/17-67
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000_0000_0000	17.4.6.2/17-69
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000_0000_0000	17.4.6.3/17-70
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000_0000_0000	17.4.6.4/17-70
0x3_4028	AESUSR—AESU status register	R	0x0000_0000_0000_0000	17.4.6.5/17-71
0x3_4030	AESUISR—AESU interrupt status register	R	0x0000_0000_0000_0000	17.4.6.6/17-72
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000_0000_1000	17.4.6.7/17-73
0x3_4050	AESUEMR—AESU end of message register	W	0x0000_0000_0000_0000	17.4.6.8/17-75
0x3_4100	AESU context memory registers	R/W	0x0000_0000_0000_0000	17.4.6.9/17-75
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000_0000_0000	17.4.6.9.5/17-79
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000_0000_0000	17.4.6.9.6/17-80
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000_0000_0000	17.4.4.1/17-51
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000_0000_0000	17.4.4.3/17-53
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000_0000_0000	17.4.4.4/17-54
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000_0000_0000	17.4.4.5/17-54
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000_0000_0000	17.4.4.6/17-55
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000_0000_0000	17.4.4.7/17-56
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000_0000_1000	17.4.4.8/17-57
0x3_6050	MDEUEUG—MDEU EU-Go register	W	0x0000_0000_0000_0000	17.4.4.9/17-58

Table 17-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000_0000_0000	17.4.4.10/17-59
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000_0000_0000	17.4.4.11/17-60
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000_0000_0000	17.4.4.12/17-60
Arc Four Execution Unit (AFEU)				
0x3_8000	AFEUMR—AFEU mode register	R/W	0x0000_0000_0000_0000	17.4.3.1/17-42
0x3_8008	AFEUKSR—AFEU key size register	R/W	0x0000_0000_0000_0000	17.4.3.3/17-43
0x3_8010	AFEUDSR—AFEU data size register	R/W	0x0000_0000_0000_0000	17.4.3.4/17-44
0x3_8018	AFEURCR—AFEU reset control register	R/W	0x0000_0000_0000_0000	17.4.3.5/17-45
0x3_8028	AFEUSR—AFEU status register	R	0x0000_0000_0000_0000	17.4.3.6/17-46
0x3_8030	AFEUISR—AFEU interrupt status register	R	0x0000_0000_0000_0000	17.4.3.7/17-47
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000_0000_1000	17.4.3.8/17-48
0x3_8050	AFEUEMR—AFEU end of message register	W	0x0000_0000_0000_0000	17.4.3.9/17-50
0x3_8100–0x3_81FF	AFEU context memory registers	R/W	0x0000_0000_0000_0000	17.4.3.10.1/17-50
0x3_8200	AFEU context memory pointers	R/W	0x0000_0000_0000_0000	17.4.3.10.2/17-51
0x3_8400	AFEUK0—AFEU key register 0	W	—	17.4.3.11/17-51
0x3_8408	AFEUK1—AFEU key register 1	W	—	17.4.3.11/17-51
0x3_8800–0x3_8FFF	AFEU FIFO	R/W	0x0000_0000_0000_0000	17.4.3.11.1/17-51
Random Number Generator (RNG)				
0x3_A000	RNGMR—RNG mode register	R/W	0x0000_0000_0000_0000	17.4.5.1/17-61
0x3_A010	RNGDSR—RNG data size register	R/W	0x0000_0000_0000_0000	17.4.5.2/17-62
0x3_A018	RNGRCR—RNG reset control register	R/W	0x0000_0000_0000_0000	17.4.5.3/17-63
0x3_A028	RNGSR—RNG status register	R	0x0000_0000_0000_0000	17.4.5.4/17-63
0x3_A030	RNGISR—RNG interrupt status register	R	0x0000_0000_0000_0000	17.4.5.5/17-64
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000_0000_1000	17.4.5.6/17-65
0x3_A050	RNGEUG—RNG EU-Go register	W	0x0000_0000_0000_0000	17.4.5.7/17-66
0x3_A800–0x3_AFFF	RNG FIFO	R	0x0000_0000_0000_0000	17.4.5.8/17-66
Public Key Execution Unit (PKEU)				
0x3_C000	PKEUMR—PKEU mode register	R/W	0x0000_0000_0000_0000	17.4.1.1/17-25
0x3_C008	PKEUKSR—PKEU key size register	R/W	0x0000_0000_0000_0000	17.4.1.2/17-26
0x3_C010	PKEUDSR—PKEU data size register	R/W	0x0000_0000_0000_0000	17.4.1.3/17-26

Table 17-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C018	PKEURCR—PKEU reset control register	R/W	0x0000_0000_0000_0000	17.4.1.5/17-28
0x3_C028	PKEUSR—PKEU status register	R	0x0000_0000_0000_0000	17.4.1.6/17-29
0x3_C030	PKEUISR—PKEU interrupt status register	R	0x0000_0000_0000_0000	17.4.1.7/17-30
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000_0000_1000	17.4.1.8/17-31
0x3_C040	PKEUABS—PKEU AB size register	R/W	0x0000_0000_0000_0000	17.4.1.3/17-26
0x3_C050	PKEUEUG—PKEU EU-Go	W	0x0000_0000_0000_0000	17.4.1.9/17-32
0x3_C200–0x3_C23F	PKEU parameter memory A0	R/W	0x0000_0000_0000_0000	17.4.1.10/17-32
0x3_C240–0x3_C27F	PKEU parameter memory A1	R/W	0x0000_0000_0000_0000	
0x3_C280–0x3_C2BF	PKEU parameter memory A2	R/W	0x0000_0000_0000_0000	
0x3_C2C0–0x3_C2FF	PKEU parameter memory A3	R/W	0x0000_0000_0000_0000	
0x3_C300–0x3_C33F	PKEU parameter memory B0	R/W	0x0000_0000_0000_0000	
0x3_C340–0x3_C37F	PKEU parameter memory B1	R/W	0x0000_0000_0000_0000	
0x3_C380–0x3_C3BF	PKEU parameter memory B2	R/W	0x0000_0000_0000_0000	
0x3_C3C0–0x3_C3FF	PKEU parameter memory B3	R/W	0x0000_0000_0000_0000	
0x3_C400–0x3_C4FF	PKEU parameter memory E	W	0x0000_0000_0000_0000	
0x3_C800–0x3_C8FF	PKEU parameter memory N	R/W	0x0000_0000_0000_0000	

17.3 Descriptor Overview

The host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. After 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. After 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.

17.3.1 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, that is, eight long words, consisting of one header dword and seven pointer dwords. See Figure 17-3 for the descriptor format.

	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 17-3. Descriptor Format

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 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 dword that is not needed can be given a length of zero, and the channel will skip 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.

17.3.2 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				

	32–63
Field	—

Figure 17-4. Header Dword

Table 17-4. Header Dword Bit Definitions

Bits	Name	Description
	OP_0:	
0–3	EU_SEL0	Primary EU select. See Section 17.3.2.1, “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 17.3.2.1, “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 17.3.2.2, “Selecting Descriptor Type—DESC_TYPE,” for possible values.
29	—	Reserved.
30	DIR	Direction—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 CDIE (channel done interrupt enable) and CDWE (channel done writeback enable) bits in the channel configuration register. When writeback is used, the value written back is 0xFF in bits 0–7 of this long word.

17.3.2.1 Selecting Execution Units—EU_SEL0 and EU_SEL1

Table 17-5 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) result 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 results in 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 result in an unrecognized header error condition.

Table 17-5. 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

17.3.2.2 Selecting Descriptor Type—DESC_TYPE

Table 17-6 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 2.0 types, which have one in the last bit (xxxx_1).

Table 17-6. 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
0000_0	aesu_ctr_nonsnoop	AESU CTR nonsnooping
0001_0	common_nonsnoop	Common, nonsnooping, non-PKEU, non-AFEU
0010_0	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_0	—	Reserved
0111_0	—	Reserved
1000_0	pkeu_mm	PKEU-Montgomery Multiplication

Table 17-6. Descriptor Types (continued)

Value (binary)	Descriptor Type	Notes
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 standard AES ccmp	CCMP encryption and hashing, suitable for 802.11i standard
0010_1	srtp	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

[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 information about descriptor types and the data used for each type, see Section 17.3.5, “Descriptor Types.”

17.3.3 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 staring 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, current usage is limited to pointer dwords 4-6.

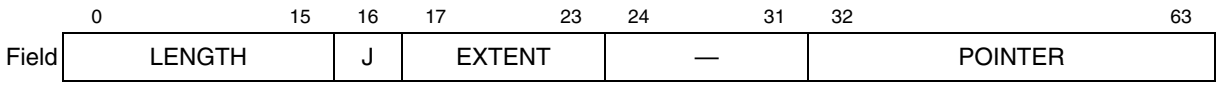


Figure 17-5. Pointer Dword

Table 17-7. Pointer Dword Field Definitions

Bits	Name	Description
0–15	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 lword 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	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.
24–27	—	Reserved
32–63	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.

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 17.3.4, “Link Table Format.”

17.3.4 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 17-7).

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 17-7.

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 (SegAdr) 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 SegAdr 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 17-7.

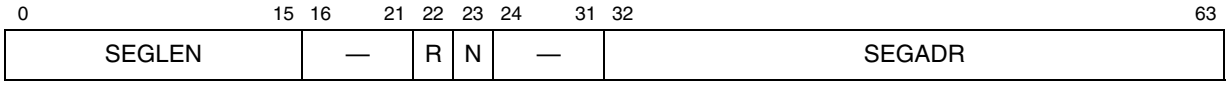


Figure 17-6. Link Table Entry Format

Table 17-8. 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 Section 17.5.1.1, “Crypto-Channel Configuration Register (CCCR)”). 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 Section 17.5.1.1, “Crypto-Channel Configuration Register (CCCR)”). Note: When N = 1, this field is ignored.
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 (SEGLEN fields). Otherwise the channel sets the appropriate error bit in the channel pointer status register—GER for gather error or SER for scatter error (see Section 17.5.1.1, “Crypto-Channel Configuration Register (CCCR)”).

17.3.4.1 Link Table Example

Figure 17-7 illustrates various ways that a descriptor may specify data parcels:

- The first pointer dword in the descriptor specifies Parcel A using the simplest method: the parcel is specified directly through Pointer0 and Length0.
- The next pointer dword 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.

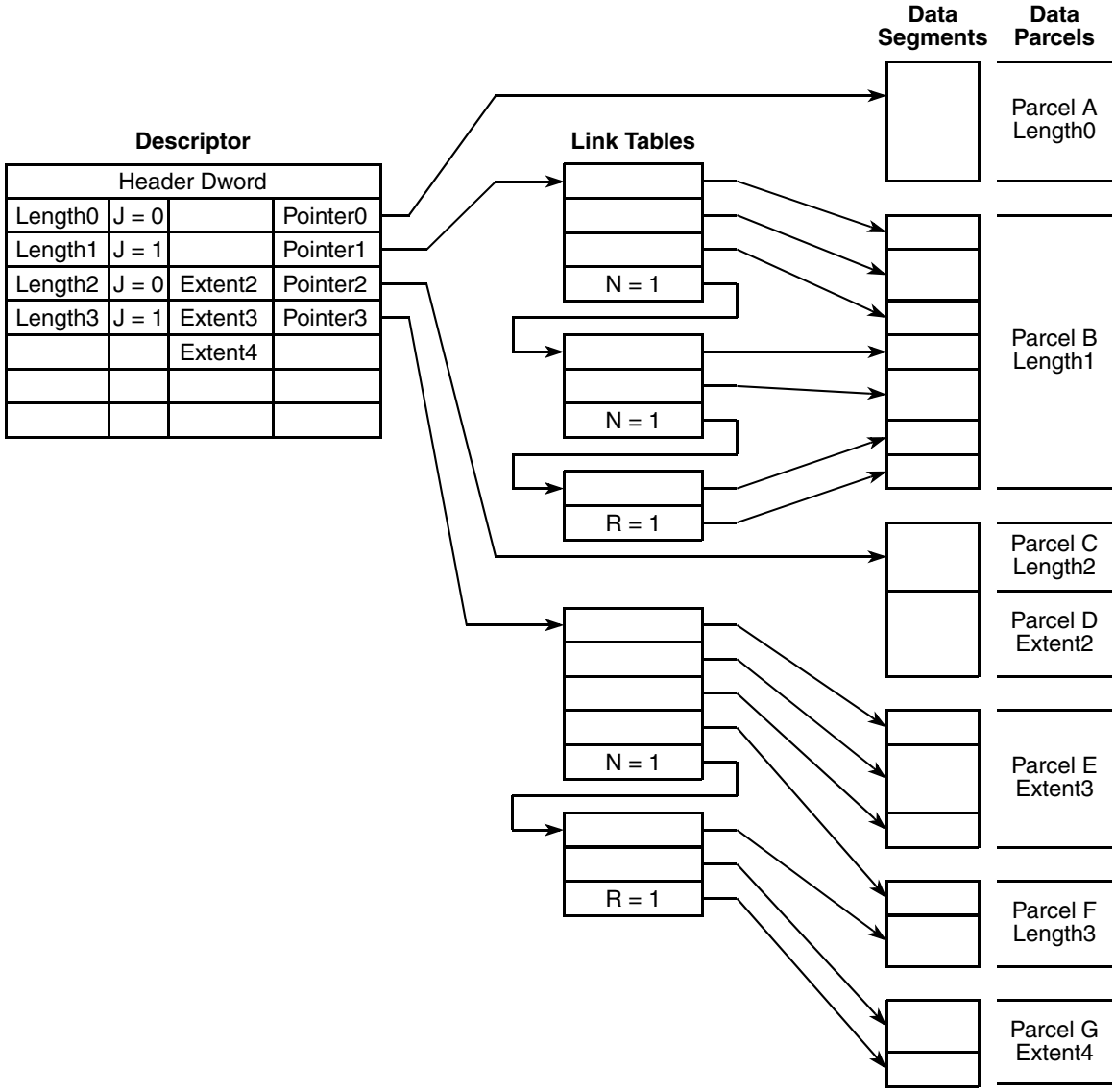


Figure 17-7. Descriptors, Link Tables, and Data Parcels

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.

17.3.5 Descriptor Types

Table 17-9 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 2.0 Descriptor Programmer's Guide*.

Table 17-9. Descriptor Pointer Dword Usage

Descriptor Type	Pointer Dword1	Pointer Dword2	Pointer Dword3	Pointer Dword4	Extent4	Pointer Dword5	Extent5	Pointer Dword6	Extent6	Pointer Dword7
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	ARC4 context (in FIFO)	ARC4 key	In FIFO	Nil	Out FIFO	Nil	ARC4 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									

Table 17-9. Descriptor Pointer Dword Usage (continued)

Descriptor Type	Pointer Dword1	Pointer Dword2	Pointer Dword3	Pointer Dword4	Extent4	Pointer Dword5	Extent5	Pointer Dword6	Extent6	Pointer Dword7
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
others	Reserved									

17.4 Execution Units

Execution unit (EU) is the term used 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 standard processing, and can work together to perform high level cryptographic tasks.

The following execution units are used in the SEC:

- Public key execution unit (PKEU)
- Data Encryption Standard execution unit (DEU)
- Advanced Encryption Standard execution unit (AESU) implementing the Rijndael symmetric-key cipher.
- ARC Four execution unit (AFEU)
- Message digest execution unit (MDEU)
- One private on-chip random number generator (RNG)

Working together, the EUs can perform high-level cryptographic tasks, such as IPSec Encapsulating Security Protocol (ESP) and digital signature. The remainder of this chapter provides details about the execution units themselves.

17.4.1 Public Key Execution Unit (PKEU)

This section contains details about the public key execution unit (PKEU), including detailed register map, modes of operation, status and control registers, and the parameter RAMs. The following sections describe PKEU registers and parameter memories.

17.4.1.1 PKEU Mode Register (PKEUMR)

This register specifies the internal PKEU routine to be executed. PKEUMR is cleared when the PKEU is reset or re-initialized. Setting the MODE field to a reserved value will generate a data error. If PKEUMR is modified during processing, a context error will be generated.

Figure 17-8 shows PKEUMR, and Table 17-10 lists the possible MODE field values.

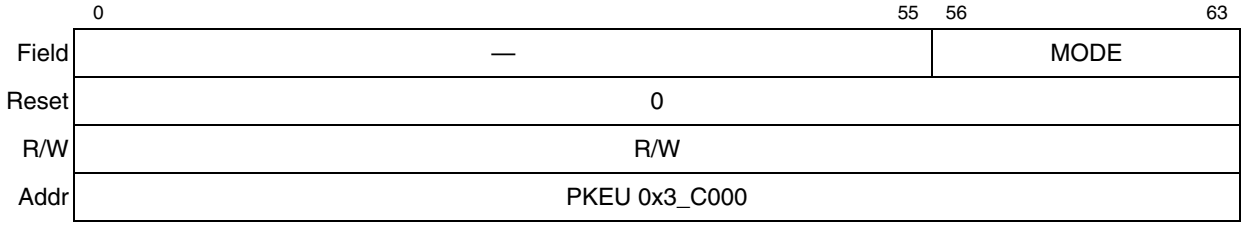


Figure 17-8. PKEU Mode Register (PKEUMR)

Table 17-10 lists the possible MODE field values. Depending on the value written to PKEUMR[MODE] will depend on the routine used. Parameter memories are referred to for the base address, as shown.

Table 17-10. PKEUMR MODE Field Descriptions

Routine Name	Routine Description	MODE
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	F_2m 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	F_2m 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	F_2m EC: Add two points in projective coordinates	0x0B
EC_F2M_DOUBLE	F_2m EC: Double a point in projective coordinates	0x0C
F2M_R2	F_2m: Compute Montgomery converter (R2 mod N)	0x0D
F2M_INV[1]	F_2m: 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

Table 17-10. PKEUMR MODE Field Descriptions (continued)

Routine Name	Routine Description	MODE
MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format	0x40
F2M_ADD	F_2m: Add mod N	0x50
F2M_MULT1_MONT	F_2m: Multiply mod N in Montgomery format	0x60
F2M_MULT2_DECONV	F_2m: 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.

17.4.1.2 PKEU Key Size Register (PKEUKSR)

The PKEU key size register, shown in Figure 17-9, 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 note 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.

	0 — 51	52 — 63
Field	—	Key Size
Reset	0	
R/W	R/W	
Addr	PKEU 0x3_C008	

Figure 17-9. PKEU Key Size Register (PKEUKSR)

17.4.1.3 PKEU AB Size Register (PKEUABS)

The PKEU AB size register (Figure 17-10) 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 will be generated as follows:

- For all non-ECC routines, a data size >256 will generate an illegal data size error.
- For all ECC routines, a data size >64 will generate an illegal data size error.

Setting PKEUABS[AB Size] = 0 (either intentionally or by ignoring it and not writing it at all) will generate an illegal size error, except for routines that do not require an A or B operand, such as the CLEAR_MEM routine.

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.

	0 — 51	52 — 63
Field	—	AB Size
Reset	0	
R/W	R/W	
Addr	PKEU 0x3_C040	

Figure 17-10. PKEU AB Size Register (PKEUABS)

17.4.1.4 PKEU Data Size Register (PKEUDSR)

The PKEU data size register, shown in Figure 17-11, specifies the size in bits of the significant portion of the modulus or irreducible polynomial. Any value written to this register that is a multiple of 32 bits (for example, 128 bits or 160 bits) will be represented internally as the same value (128 bits or 160 bits). Any value written that is not a multiple of 32 bits (for example, 132 bits or 161 bits) will be represented internally as the next larger 32 bit multiple (160 bits or 196 bits). This internal rounding up to the next 32-bit multiple is described for information only. The minimum size valid for all routines to operate properly is 97 bits (internally 128 bits). The maximum size to operate properly is 2048 bits. A value in bits larger than 2048 will result in a data size error.

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.

	0 — 51	52 — 63
Field	—	Data Size
Reset	0	
R/W	R/W	
Addr	PKEU 0x3_C010	

Figure 17-11. PKEU Data Size Register (PKEUDSR)

17.4.1.5 PKEU Reset Control Register (PKEURCR)

This register, shown in Figure 17-12, contains three reset options specific to the PKEU.

	0 — 60	61	62	63
Field	—	RI	MI	SR
Reset	0			
R/W	R/W			
Addr	PKEU 0x3_C018			

Figure 17-12. PKEU Reset Control Register (PKEURCR)

Table 17-11 describes PKEURCR fields.

Table 17-11. 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 17.4.1.6, “PKEU Status Register (PKEUSR)”). 0 Don’t reset. 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 17.4.1.6, “PKEU Status Register (PKEUSR).”) 0 Don’t reset. 1 Full PKEU reset.

17.4.1.6 PKEU Status Register (PKEUSR)

This status register contains fields which reflect the state of PKEU internal fields.

PKEUSR is read only. Writing to this register will result in an address error being reflected in the PKEU interrupt status register (PKEUISR[AE] is set).

	0–56	57	58	59–60	61	62	63
Field	—	Z	HALT	—	IE	ID	RD
Reset	0						
R/W	Read Only						
Addr	PKEU 0x3_C028						

Figure 17-13. PKEU Status Register (PKEUSR)

Table 17-12 describes PKEUSR fields.

Table 17-12. PKEUSR 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 Z to be modified, so this bit should be used with great care.
58	HALT	Halt indicates that 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.
59–60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (See Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 PKEU is not signaling error 1 PKEU is signaling error
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (See Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 PKEU is not signaling done 1 KEU is signaling done
63	RD	Reset done. This status bit, when high, indicates that PKEU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

17.4.1.7 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 17-14.

	0–49	50	51	52	53	54	55	56	57	58–63
Field	—	INV	IE	—	CE	KSE	DSE	ME	AE	—
Reset	0									
R/W	Read only									
Addr	PKEU 0x3_C030									

Figure 17-14. PKEU Interrupt Status Register (PKEUISR)

Table 17-13 describes PKEUISR fields.

Table 17-13. 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. PKEUKR*n*, PKEUKSR, PKEUDSR, or PKEUMR was modified while the PKEU was operating. 0 No error detected 1 Context error
54	KSE	Key size error. Value outside the bounds of 1–256 bytes was written to PKEUKSR 0 No error detected 1 Key size error detected
55	DSE	Data size error. Value outside the bounds 97–2048 bits was written to PKEUDSR 0 No error detected 1 Data size error detected
56	ME	Mode error. An illegal value was detected in PKEUMR. 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

17.4.1.8 PKEU Interrupt Control Register (PKEUICR)

The PKEU interrupt control register controls the result of detected errors. For a given error (as defined in Section 17.4.1.7, “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.

	0–49	50	51	52	53	54	55	56	57	58–63
Field	—	INV	IE	—	CE	KSE	DSE	ME	AE	—
Reset	0x0000_0000_0000_1000									
R/W	Read only									
Addr	PKEU 0x3_C038									

Figure 17-15. PKEU Interrupt Control Register (PKEUICR)

Table 17-14 describes PKEUICR fields.

Table 17-14. 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

17.4.1.9 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.

	0	63
Field	PKEU EU-Go	
Reset	0	
R/W	W	
Addr	PKEU 0x3_C050	

Figure 17-16. PKEU EU-Go Register (PKEUEUG)

17.4.1.10 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.

17.4.1.10.1 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.

17.4.1.10.2 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.

17.4.1.10.3 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.

17.4.1.10.4 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.

17.4.2 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.

The registers used in the DEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the DEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and on-chip controller will abstract register-level access from the user.

17.4.2.1 DEU Mode Register (DEUMR)

The DEU mode register contains 3 bits which are used to program the DEU. 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.

DEUMR is cleared when the DEU 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.

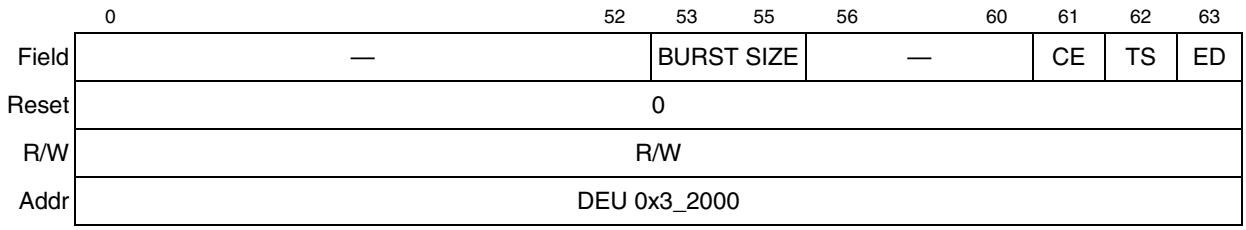


Figure 17-17. DEU Mode Register (DEUMR)

Table 17-15 describes DEUMR fields.

Table 17-15. DEUMR Field Descriptions

Bits	Name	Description
0–52	—	Reserved
53–55	BURST SIZE	Implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/IV. The DEU signals to the channel that a BURST SIZE amount of data is available to be pushed to or pulled from the FIFO. **Note:** The inclusion of this field in DEUMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the DEU.

Table 17-15. DEUMR Field Descriptions (continued)

Bits	Name	Description
56–60	—	Reserved
61	CE	CBC/ECB. If set, DEU operates in cipher-block-chaining mode. If not set, DEU operates in electronic codebook mode. 0 ECB mode 1 CBC mode
62	TS	Triple/Single DES. If set, DEU performs the triple DES algorithm; if not set, DEU performs the single DES algorithm. 0 Perform single DES. 1 Perform triple DES.
63	ED	Encrypt/Decrypt. If set, DEU performs the encryption algorithm; if not set, DEU performs the decryption algorithm. 0 Perform decryption. 1 Perform encryption.

17.4.2.2 DEU Key Size Register (DEUKSR)

This value indicates the number of bytes of key memory that should be used in encrypting or decrypting. If DEUMR is set for single DES, any value other than 8 bytes will automatically generate 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) will generate 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.

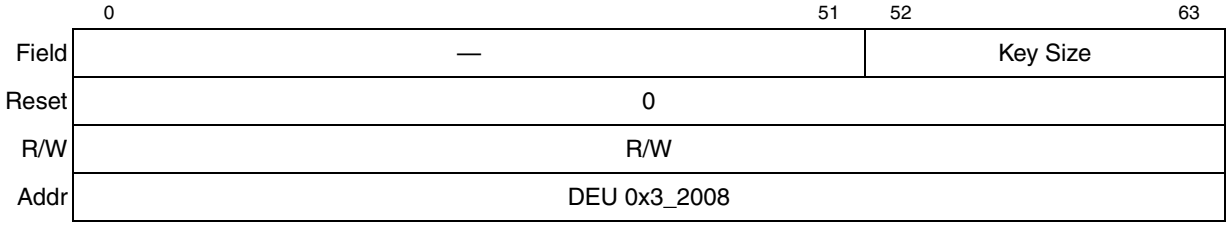


Figure 17-18. DEU Key Size Register (DEUKSR)

Table 17-16 shows the legal values for DEUKSR[Key Size].

Table 17-16. 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)

17.4.2.3 DEU Data Size Register (DEUDSR)

This register, shown in Figure 17-19, 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.

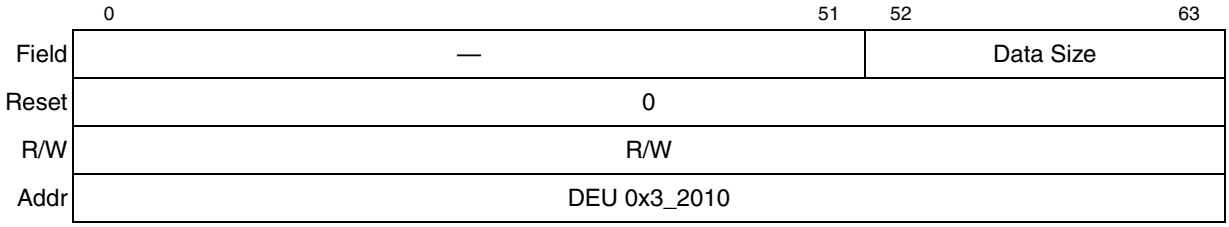


Figure 17-19. DEU Data Size Register (DEUDSR)

17.4.2.4 DEU Reset Control Register (DEURCR)

This register, shown in Figure 17-20, allows three levels of reset of just the DEU, as defined by the three self-clearing bits:

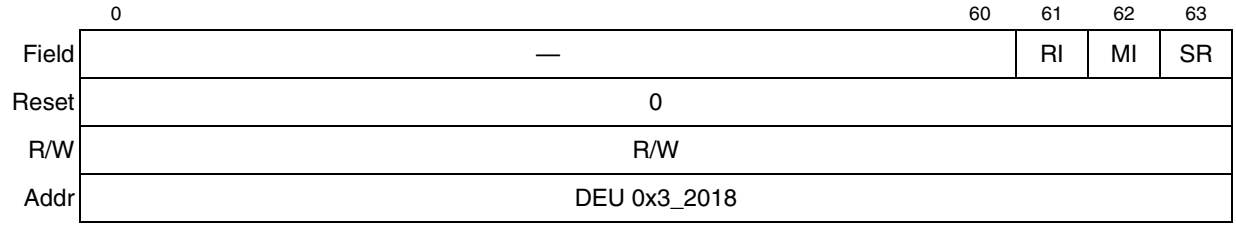


Figure 17-20. DEU Reset Control Register (DEURCR)

Table 17-17 describes DEURCR fields.

Table 17-17. 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

17.4.2.5 DEU Status Register (DEUSR)

The DEU status register, displayed in Figure 17-21, contains six fields which reflect the state of DEU internal signals. DEUSR is read only; writing to DEUSR will result in an address error being reflected in DEUISR.

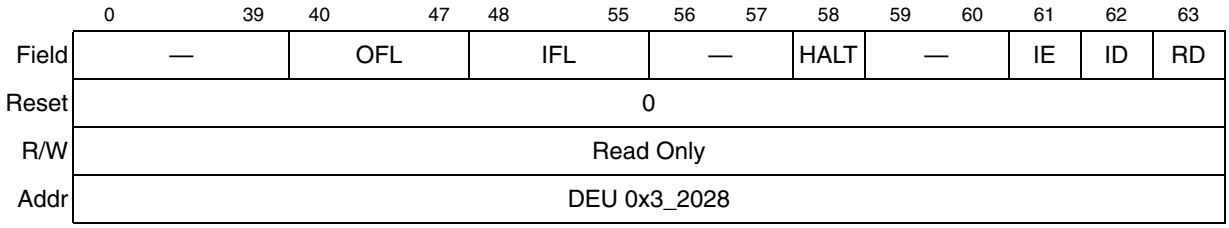


Figure 17-21. DEU Status Register (DEUSR)

Table 17-12 describes DEUSR fields.

Table 17-18. DEUSR Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the DEU has halted due to an error. 0 DEU not halted 1 DEU halted **Note:** Because the error causing the DEU 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.
59–60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 DEU is not signaling error. 1 DEU is signaling error.
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 DEU is not signaling done. 1 DEU is signaling done.
63	RD	Reset done. This status bit, when high, indicates that DEU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

17.4.2.6 DEU Interrupt Status Register (DEUISR)

The DEU interrupt status register, shown in Figure 17-22, tracks the state of possible errors, if those errors are not masked through DEUICR.

	0–49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
Reset	0														
R/W	Read only														
Addr	DEU 0x3_2030														

Figure 17-22. DEU Interrupt Status Register (DEUISR)

Table 17-19 describes DEUISR fields.

Table 17-19. 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 DEUK2 and DEUK3 are checked for parity only if the appropriate DEUMR bit indicates triple DES. Also, key register 3 is checked only if DEUKSR[Key Size] = 24. DEUK2 is checked only if DEUKSR[Key Size] = 16 or 24.) 0 No error detected 1 Key parity error detected
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error detected **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 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 detected
53	CE	Context error. DEUKR, DEUKSR, DEUDSR, DEUMR, or DEUIV was modified while the 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 DEUKSR. 0 No error detected 1 Key size error detected
55	DSE	Data size error (DSE). A value was written to DEUDSR that is not a multiple of 64 bits. 0 No error detected 1 Data size error detected
56	ME	Mode error. An illegal value was detected in DEUMR. Note: writing to reserved bits in DEUMR is the likely source of error. 0 No error detected 1 Mode error

Table 17-19. DEUISR Field Descriptions (continued)

Bits	Name	Description
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 DEUDSR. 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 had underflow error
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 DEU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the DEU 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

17.4.2.7 DEU Interrupt Control Register (DEUICR)

The DEU interrupt control register controls the result of detected errors. For a given error (as defined in Section 17.4.2.6, “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.

	0–49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
Reset	0x0000_0000_0000_3000														
R/W	R/W														
Addr	DEU 0x3_2038														

Figure 17-23. DEU Interrupt Control Register (DEUICR)

Table 17-20. 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 DEUK2 and DEUK3 are only checked for parity if the appropriate DEUMR bit indicates triple DES. 0 Key parity error 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
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. DEUKR, DEUKSR, DEUDSR, DEUMR, or DEUIV was modified while the 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 DEUKSR. 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error (DSE): A value was written to DEUDSR 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 DEUMR. 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 error. 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 error. 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 DEU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the DEU cannot accept FIFO inputs larger than 512 bytes without overflowing.

Table 17-20. DEUICR Field Descriptions (continued)

Bits	Name	Description
62	OFU	Output FIFO underflow error. 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

17.4.2.8 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.

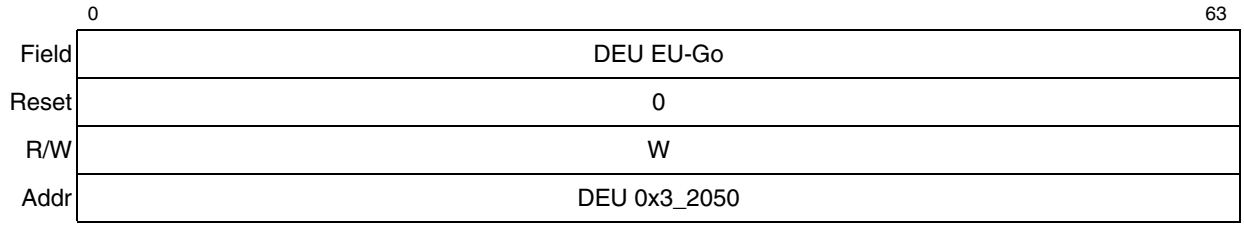


Figure 17-24. DEU EU-Go Register (DEUEUG)

17.4.2.9 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.

17.4.2.10 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.

17.4.2.11 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.

17.4.3 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.

17.4.3.1 AFEU Mode Register (AFEUMR)

Shown in Figure 17-25, 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.

17.4.3.2 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.

17.4.3.2.1 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.

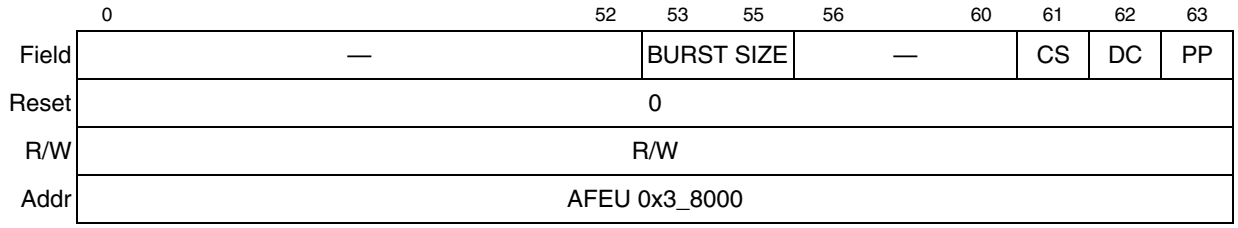


Figure 17-25. AFEU Mode Register (AFEUMR)

Table 17-21 describes AFEU mode register fields.

Table 17-21. AFEUMR Field Descriptions

Bits	Name	Description
0–52	—	Reserved
53–55	BURST SIZE	The AFEU implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/context. The AFEU signals to the channel that BURST SIZE amount of data is available to be pushed to or pulled from the FIFO. **Note:** The inclusion of this field in the AFEUMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the AFEU.
56–60	—	Reserved
61	CS	Context source. If set, CS causes the context to be moved from the input FIFO into the S-box prior to starting encryption/decryption. Otherwise, context should be directly written to the context registers. CS is only checked if PP is set. 0 Context not from FIFO 1 Context from input FIFO
62	DC	Dump context. If set, this causes the context to be moved from the S-box to the output FIFO following assertion AFEU's done interrupt. 0 Do not dump context 1 After cipher, dump context
63	PP	Prevent permute. Normally, AFEU receives a key and uses that information to randomize the S-box. If reusing a context from a previous descriptor, PP should be set to prevent AFEU from reperforming this permutation step. 0 Perform S-Box permutation 1 Do not permute

17.4.3.3 AFEU Key Size Register (AFEUKSR)

As displayed in Figure 17-26, this value indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in AFEUKSR will be ignored.

AFEUKSR is cleared when the AFEU is reset or re-initialized. If the key size specified is less than 1 or greater than 16, a key size error will be generated. If AFEUKSR is modified during processing, a context error will be generated. Note that although the AFEU supports key lengths as short as 1 byte, a 1-byte key offers little security. Most uses of ARC4 specify keys of 5–16 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 prior to performing debug operations.

	0 — 51	52 — 63
Field	—	Key Size
Reset	0	
R/W	R/W	
Addr	AFEU 0x3_8008	

Figure 17-26. AFEU Key Size Register (AFEUKSR)

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.

17.4.3.4 AFEU Context/Data Size Register (AFEUDSR)

The AFEU context/data size register, shown in Figure 17-27, 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.

	0 — 51	52 — 63
Field	—	Data Size
Reset	0	
R/W	R/W	
Addr	AFEU 0x3_8010	

Figure 17-27. AFEU Data Size Register (AFEUDSR)

17.4.3.5 AFEU Reset Control Register (AFEURCR)

This register, shown in Figure 17-28, 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.

	0 — 60	61	62	63
Field	—	RI	MI	SR
Reset	0			
R/W	R/W			
Addr	AFEU 0x3_8018			

Figure 17-28. AFEU Reset Control Register (AFEURCR)

Table 17-22 describes AFEURCR fields.

Table 17-22. 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 will enter a routine to perform proper initialization of the S-Box. 0 Do not reset 1 Full AFEU reset

NOTE

The AFEU must be reset prior to first use following a power on reset. Failure to do so results in undefined behavior. Set the SR bit to perform AFEU reset.

17.4.3.6 AFEU Status Register (AFEUSR)

This status register, shown in Figure 17-29, contains 6 bits which reflect the state of the AFEU internal signals.

The AFEUSR is read-only. Writing to this location will result in address error being reflected in AFEUISR.

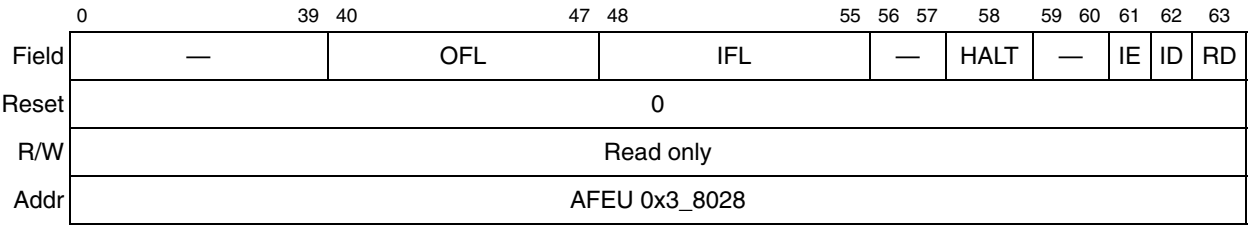


Figure 17-29. AFEU Status Register (AFEUSR)

Table 17-23 describes AFEUSR fields.

Table 17-23. AFEUSR Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO

Table 17-23. AFEUSR Field Descriptions (continued)

Bits	Name	Description
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the AFEU has halted due to an error. 0 AFEU not halted 1 AFEU halted **Note:** Because the error causing the AFEU 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.
59–60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 AFEU is not signaling error. 1 AFEU is signaling error.
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 AFEU is not signaling done. 1 AFEU is signaling done.
63	RD	Reset done. This status bit, when high, indicates that AFEU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

17.4.3.7 AFEU Interrupt Status Register (AFEUISR)

The AFEU interrupt status register, seen in Figure 17-30, tracks the state of possible errors, if those errors are not masked through AFEUICR.

	0 — 50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	0													
R/W	Read only													
Addr	AFEU 0x3_8030													

Figure 17-30. AFEU Interrupt Status Register (AFEUISR)

Table 17-24 describes AFEUISR fields.

Table 17-24. 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

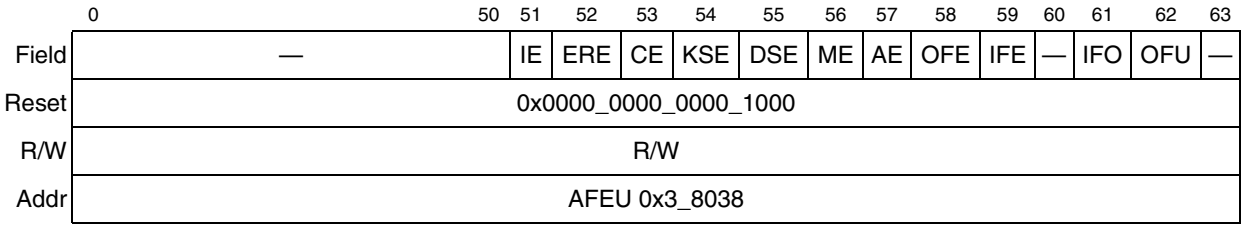
Table 17-24. AFEUISR Field Descriptions (continued)

Bits	Names	Description
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

17.4.3.8 AFEU Interrupt Control Register (AFEUICR)

The interrupt control register, shown in Figure 17-31, controls the result of detected errors. For a given error (as defined in Section 17.4.3.7, “AFEU Interrupt Status Register (AFEUISR)”), if the corresponding bit in AFEUICR is set, the error is disabled; no error interrupt occurs, and AFEUISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, AFEUISR is updated

to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.



	0–50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	0x0000_0000_0000_1000													
R/W	R/W													
Addr	AFEU 0x3_8038													

Figure 17-31. AFEU Interrupt Control Register (AFEUICR)

Table 17-25 describes AFEUICR fields.

Table 17-25. AFEUICR Field Descriptions

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The AFEU register was read while the AFEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. AFEUKR*n*, AFEUKSR, AFEUDSR, AFEUMR, or context memory was modified while AFEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. A value outside the bounds 1–16 bytes was written to AFEUKSR. 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value was written to the AFEUDSR. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in AFEUMR. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 Address error enabled 1 Address error disabled
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

Table 17-25. AFEUICR Field Descriptions (continued)

Bits	Names	Description
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

17.4.3.9 AFEU End of Message Register (AFEUEMR)

The end of message register in the AFEU, displayed in Figure 17-32, 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 AFEUMR[DC] is set (dump context mode is enabled), the context will be written to the output FIFO following the last message word. A read of AFEUEMR will always return 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.

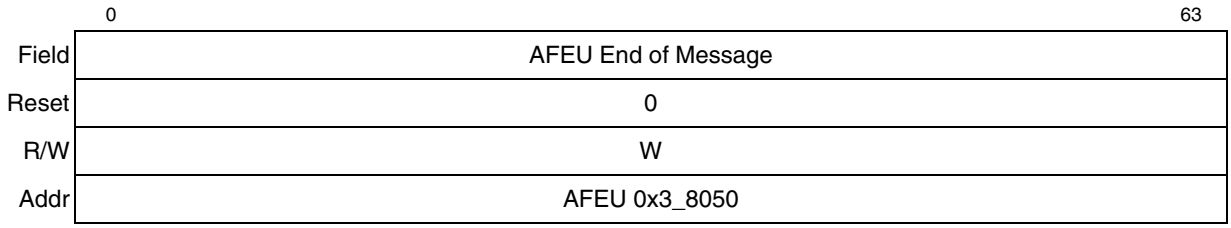


Figure 17-32. AFEU End of Message Register (AFEUEMR)

17.4.3.10 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

17.4.3.10.1 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 17-25), 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.

17.4.3.10.2 AFEU Context Memory Pointer Registers

The context memory pointer registers hold 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.

17.4.3.11 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 the two key registers as necessary to complete the permutation. Reading either of these memory locations will generate an address error interrupt.

17.4.3.11.1 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.

17.4.4 Message Digest Execution Unit (MDEU)

This section contains details about the message digest execution unit (MDEU), including detailed register map, modes of operation, status and control registers, and FIFOs.

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.

17.4.4.1 MDEU Mode Register (MDEUMR)

The MDEU mode register, shown in Figure 17-33, contains 8 bits which are used to program the MDEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the SEC 2.0 as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

MDEUMR is cleared when the MDEU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If MDEUMR is modified during processing, a context error will be generated.

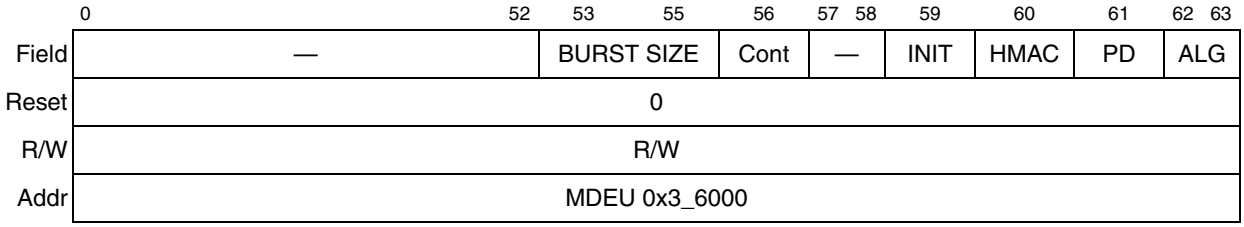


Figure 17-33. MDEU Mode Register (MDEUMR)

Table 17-26 describes MDEUMR fields.

Table 17-26. MDEUMR Field Descriptions

Bits	Name	Description
0–52	—	Reserved
53–55	BURST SIZE	The MDEU implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/context. The MDEU signals to the channel that a BURST SIZE amount of data is available to be pushed to the FIFO. **Note:** The inclusion of this field in the MDEUMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the MDEU.
56	Cont	Continue—Used during HMAC/HASH processing when the data to be hashed is spread across multiple descriptors. 0 Don't continue—operate the MDEU in auto completion mode 1 Preserve context to operate the MDEU in continuation mode
57–58	—	Reserved, set to zero.
59	INIT	Initialization bit—Cause an algorithm-specific initialization of the digest registers. Most operations will require this bit to be set. Only operations that load context from a known intermediate hash value would not initialize the registers. 0 Do not initialize. 1 Initialize the selected algorithm's starting registers.
60	HMAC	Identifies the hash operation to execute 0 Perform standard hash. 1 Perform HMAC operation. This requires a key and key length information.
61	PD	If set, configures the MDEU to automatically pad partial message blocks. 0 Do not autopad. 1 Perform automatic message padding whenever an incomplete message block is detected.
62–63	ALG	Message digest algorithm selection 00 SHA-160 algorithm (full name for SHA-1) 01 SHA-256 algorithm 10 MD5 algorithm 11 Reserved

17.4.4.2 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 SSL/TLS. When the HMAC is being generated by a single descriptor (the MDEU acting as sole or secondary EU), the following MDEUMR bit settings should be used:

Table 17-27. MDEUMR—HMAC Generated by Single Descriptor

Bits	Field	Value
56	Cont	0 (off)
59	INIT	1 (on)
60	HMAC	1 (on)
61	PD	1 (on)

The SEC cannot calculate an HMAC across multiple descriptors. However, it is capable of calculating a simple hash across multiple descriptors. When the hash is being generated for a message that is spread across a chain of descriptors, the following MDEUMR bit settings should be used:

Table 17-28. MDEUMR—HMAC Generated for a Message Across a Chain 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	0 (off)	0 (off)	0 (off)
61	PD	0 (off)	0 (off)	1 (on)

All descriptors other than the final descriptor must output the intermediate message digest for the following descriptor to reload as MDEU context.

Additional information on descriptors can be found in Section 17.1.1, “Data Packet Descriptors.”

17.4.4.3 MDEU Key Size Register (MDEUKSR)

Displayed in Figure 17-34, MDEUKSR 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 prior to performing debug operations.

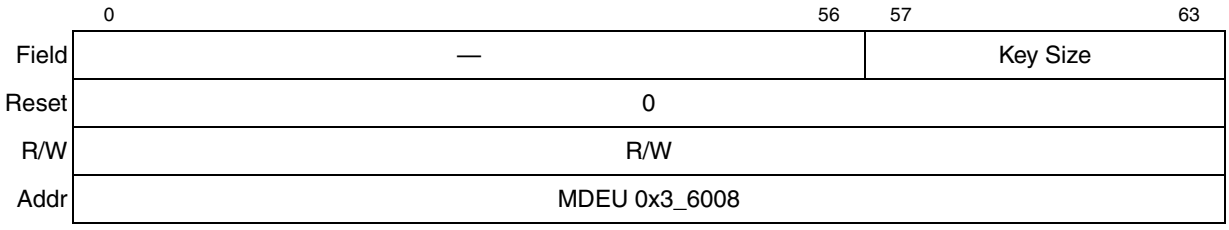


Figure 17-34. MDEU Key Size Register (MDEUKSR)

17.4.4.4 MDEU Data Size Register (MDEUDSR)

The MDEU data size register, shown in Figure 17-35, stores the size of the last block of data (in bits) to be processed. Because 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 will allow the MDEU to enter auto-start mode. Therefore, the required context data should be written prior to writing the data size.

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.

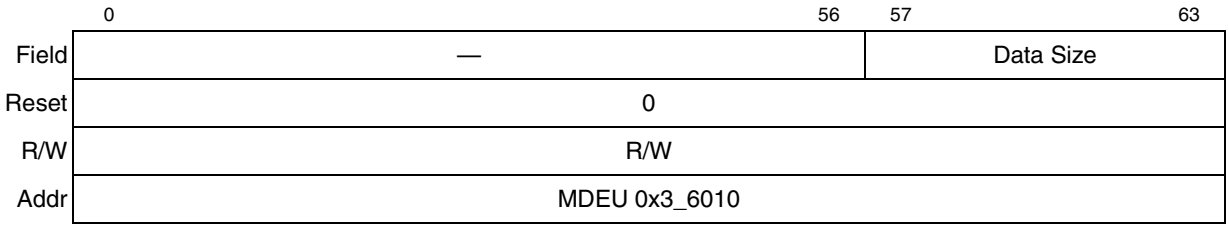


Figure 17-35. MDEU Data Size Register (MDEUDSR)

17.4.4.5 MDEU Reset Control Register (MDEURCR)

This register, shown in Figure 17-36, allows three levels of reset of just the MDEU, as defined by the three self-clearing bits.

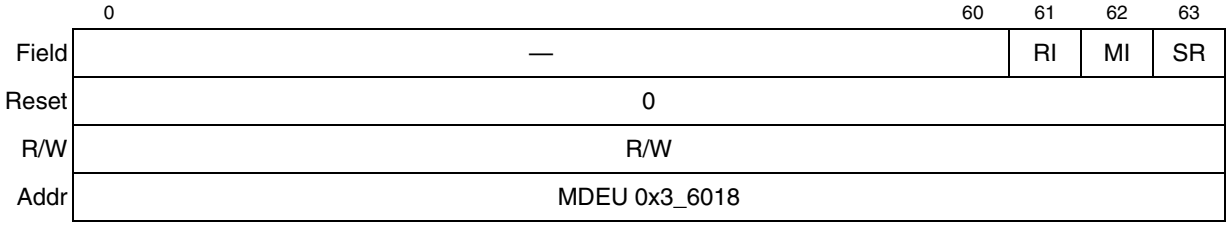


Figure 17-36. MDEU Reset Control Register (MDEURCR)

Table 17-29 describes MDEURCR fields.

Table 17-29. MDEURCR 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
62	MI	Module initialization is nearly the same as software reset, except that the MDEUICR remains unchanged. 0 No reset 1 Reset most of MDEU
63	SR	Software reset is functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ 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

17.4.4.6 MDEU Status Register (MDEUSR)

The MDEU status register, as seen in Figure 17-37, 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 which are most likely to be of interest to the user.

MDEUSR is read only.

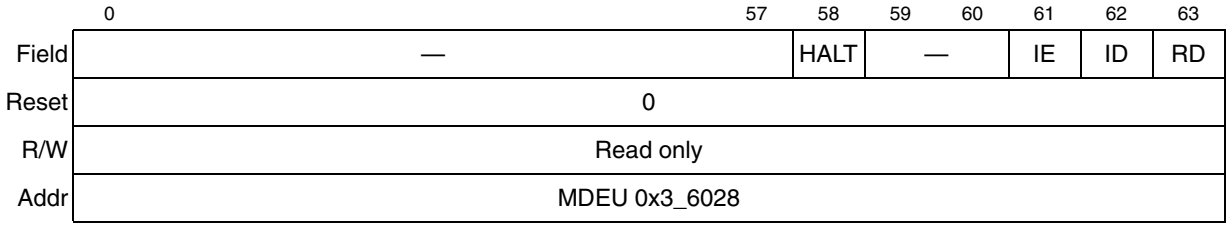


Figure 17-37. MDEU Status Register (MDEUSR)

Table 17-23 describes MDEUSR fields.

Table 17-30. MDEUSR Field Descriptions

Bits	Name	Description
0–57	—	Reserved
58	HALT	Halt. Indicates that the MDEU has 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	—	Reserved

Table 17-30. MDEUSR 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 17.6.2.2, “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 17.6.2.2, “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

17.4.4.7 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 17-38.

	0–50	51	52	53	54	55	56	57	58–60	61	62–63
Field	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—
Reset	0										
R/W	Read only										
Addr	MDEU 0x3_6030										

Figure 17-38. MDEU Interrupt Status Register (MDEUISR)

Table 17-31 describes MDEUISR fields.

Table 17-31. MDEUISR Field Descriptions

Bits	Name	Description
0–50	—	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 will be asserted 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
53	CE	Context error. The MDEU key register, MDEUKSR, or MDEUDSR was modified while MDEU was hashing. 0 No error detected 1 Context error
54	KSE	Key size error. A value greater than 64 bytes was written to MDEUKSR. 0 No error detected 1 Key size error

Table 17-31. MDEUISR Field Descriptions (continued)

Bits	Name	Description
55	DSE	Data size error. A value not a multiple of 512 bits while the MDEU mode register autopad bit is negated. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value for ALG was detected in MDEUMR. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 No error detected 1 Address error
58–60	—	Reserved
61	IFO	Input FIFO Overflow. The MDEU input FIFO has been pushed while full. 0 No overflow 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.
62–63	—	Reserved

17.4.4.8 MDEU Interrupt Control Register (MDEUICR)

The MDEU interrupt control register, shown in Figure 17-39, controls the result of detected errors. For a given error (as defined in Section 17.4.4.7, "MDEU Interrupt Status Register (MDEUISR)"), if the corresponding bit in MDEUICR is set, the error is disabled; no error interrupt occurs and MDEUISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, MDEUISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0–50	51	52	53	54	55	56	57	58–60	61	62–63
Field	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—
Reset	0x0000_0000_0000_1000										
R/W	R/W										
Addr	MDEU 0x3_6038										

Figure 17-39. MDEU Interrupt Control Register (MDEUICR)

Table 17-31 describes MDEUICR fields.

Table 17-32. MDEUICR Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing hashing. 0 Internal error enabled 1 Internal error disabled

Table 17-32. MDEUICR Field Descriptions (continued)

Bits	Name	Description
52	ERE	Early read error. The MDEU register was read while the MDEU was performing hashing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. The MDEU key register, MDEUKSR, MDEUDSR, or MDEUMR was modified while the MDEU was performing hashing. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. A value outside the bounds of 64 bytes was 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 was written to MDEUDSR: 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in MDEUMR. 0 Mode error enabled 1 Mode error disabled
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
58–60	—	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

17.4.4.9 MDEU EU-Go Register (MDEUEUG)

The EU-Go register in the MDEU (see Figure 17-40) is used to indicate that an authentication operation may be completed. After the final message block is written to the input FIFO, MDEUEUG must be written. The value in MDEUDSR will be used to determine how many bits of the final message block (always 512) will be processed. Note that MDEUEUG 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 MDEUEUG is meaningful, but no error is generated, and a zero value is always returned. Writing to MDEUEUG is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal DONE.

MDEUEUG is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the MDEU (by means of an internally generated write to MDEUEUG) when the SEC acts as an initiator.

	0	63
Field	MDEU EU-Go	
Reset	0	
R/W	W	
Addr	MDEU 0x3_6050	

Figure 17-40. MDEU EU-Go Register (MDEUEUG)

17.4.4.10 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provides the resulting message digest or HMAC, along with an aggregate bit count.

NOTE

SHA-1and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU mode register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location 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 17-41 shows how the MDEU context registers are initialized if the INIT bit is set in MDEUMR. All registers are initialized, regardless of mode selected; however, only the appropriate MDEU context register values are used in hash generation according to the mode selected. The user typically does not 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 through MDEURCR (see Figure 17-36) or SEC global software reset (see Figure 17-68) 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 17-41. MDEU Context Registers

17.4.4.11 MDEU Key Registers

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required.

NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if MDEUMR indicates MD5 is the hash of choice.

17.4.4.12 MDEU FIFOs

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 MDEUMR indicates MD5 is the hash of choice.

17.4.5 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 RNG are documented primarily for debug and slave mode operations. If the SEC requires the use of the RNG 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.

17.4.5.1 RNG Mode Register (RNGMR)

The RNG mode register is used to control the RNG. One operational mode, randomizing, is defined. Writing any other value than 0 to bits 56–63 results in a data error interrupt that is reflected in the RNGISR. RNGMR also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the RNG as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

RNGMR is cleared when the RNG is reset or re-initialized. RNGMR is shown in Figure 17-42.

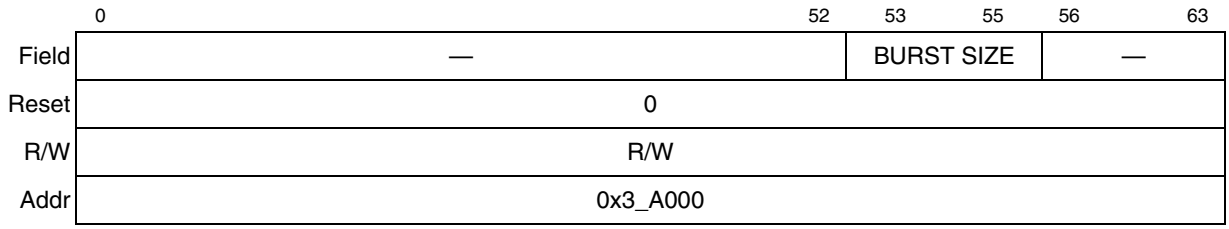


Figure 17-42. RNG Mode Register (RNGMR)

Table 17-33. RNGMR Field Definitions

Bits	Name	Description
0–52	—	Reserved, must be set to zero.
53–55	BURST SIZE	The RNG 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 RNGMR 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

17.4.5.2 RNG Data Size Register (RNGDSR)

The RNG data size register is used to tell the RNG to begin generating random data. The actual contents of RNGDSR do not affect the operation of the RNG. After a reset and prior to the first write of data size, the RNG builds entropy without pushing data onto the FIFO. Once RNGDSR 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.

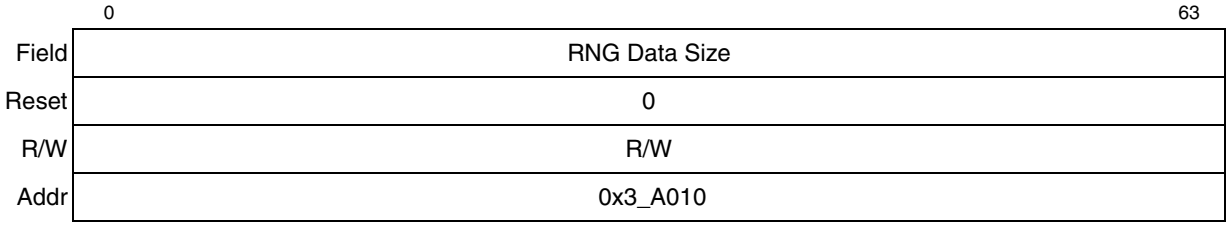


Figure 17-43. RNG Data Size Register (RNGDSR)

17.4.5.3 RNG Reset Control Register (RNGRCR)

This register, shown in Figure 17-44, contains three reset options specific to the RNG.

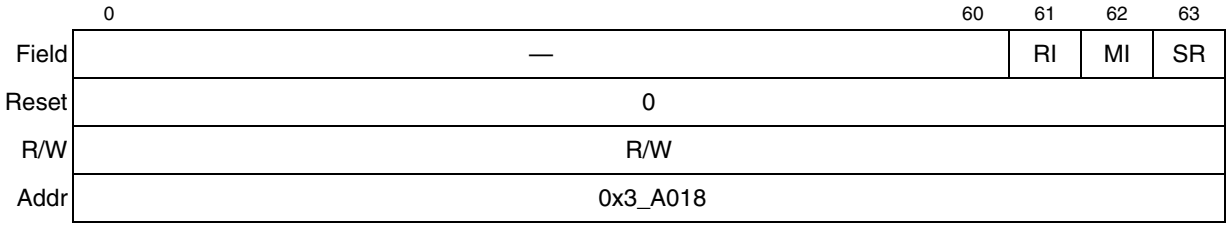


Figure 17-44. RNG Reset Control Register (RNGRCR)

Table 17-34 describes RNGRCR fields.

Table 17-34. RNGRCR Field Descriptions

Bits	Name	Description
0-60	—	Reserved
61	RI	Reset interrupt. Writing this bit causes RNG interrupts signaling DONE and ERROR to be reset. It further resets the state of the RNGISR. 0 No reset 1 Reset interrupt logic
62	MI	Module initialization. This reset value performs enough of a reset to prepare the RNG for another request, without forcing the internal control machines and the output FIFO to be reset, thereby invalidating stored random numbers or requiring reinvocation of a warm-up period. Module initialization is nearly the same as software reset, except that RNGICR remains unchanged. 0 No reset 1 Reset most of the RNG
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the RNG. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full RNG reset
8–63	—	Reserved

17.4.5.4 RNG Status Register (RNGSR)

This RNG status register, Figure 17-45, contains six fields which reflect the state of the RNG internal signals.

RNGSR is read only. Writing to this location will result in an address error being reflected in RNGISR.

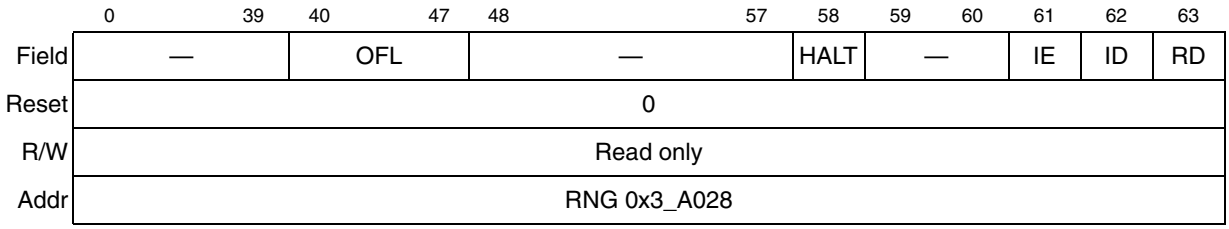


Figure 17-45. RNG Status Register (RNGSR)

Table 17-23 describes RNG status register fields.

Table 17-35. RNGSR Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40-47	OFL	The number of dwords currently in the output FIFO
48-57	—	Reserved. Internal status bits may be observed as nonzero.
58	HALT	Halt. Indicates that the RNG has halted due to an error. 0 RNG not halted 1 RNG halted **Note:** Because the error causing the RNG 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.
59-60	—	Reserved
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller ISR (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 RNG is not signaling error. 1 RNG is signaling error.
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 RNG is not signaling done. 1 RNG is signaling done.
63	RD	Reset done. This status bit, when set, indicates that the RNG has completed its internal reset sequence. 0 Reset in progress 1 Reset done

17.4.5.5 RNG Interrupt Status Register (RNGISR)

The RNG interrupt status register tracks the state of possible errors, if those errors are not masked, through the RNGICR. The definition of each bit in RNGISR is shown in Figure 17-46.

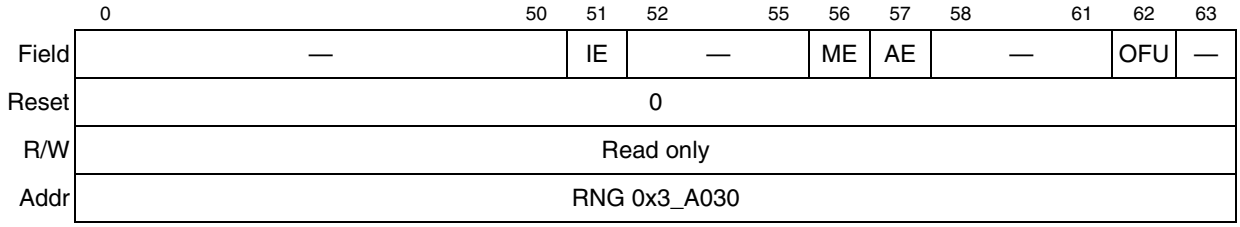


Figure 17-46. RNG Interrupt Status Register (RNGISR)

Table 17-36 describes RNGISR fields.

Table 17-36. RNGISR Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error 0 No internal error detected 1 Internal error
52–55	—	Reserved
56	ME	Mode error. Indicates that the host has attempted to write an illegal value to RNGMR. 0 Valid data 1 Invalid data error
57	AE	Address error. An illegal read or write address was detected within the RNG address space. 0 No error detected 1 Address error
58–61	—	Reserved
62	OFU	Output FIFO underflow. The RNG output FIFO has been read while empty. 0 No overflow detected 1 Output FIFO has underflowed
63	—	Reserved

17.4.5.6 RNG Interrupt Control Register (RNGICR)

The RNG interrupt control register controls the result of detected errors. For a given error (as defined in Section 17.4.5.5, “RNG Interrupt Status Register (RNGISR)”), if the corresponding bit in RNGICR is set, then the error is disabled; no error interrupt occurs and RNGISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, RNGISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

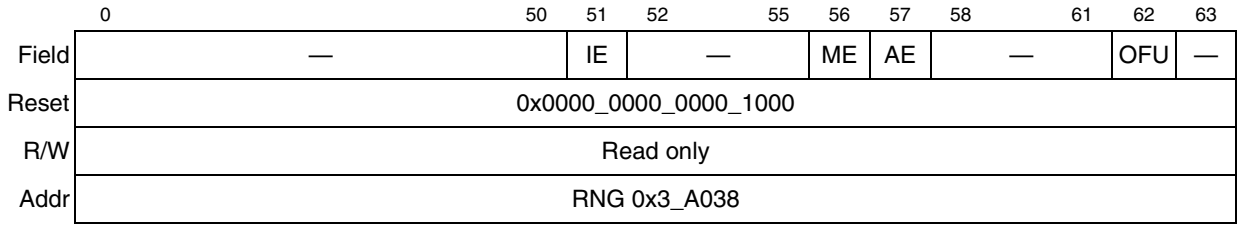


Figure 17-47. RNG Interrupt Control Register (RNGICR)

Table 17-37 describes RNGICR fields.

Table 17-37. RNGICR Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while generating random numbers. 0 Internal error enabled 1 Internal error disabled
52–55	—	Reserved
56	ME	Mode error. An illegal value was detected in RNGMR. 0 Mode error enabled 1 Mode error disabled
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

17.4.5.7 RNG EU-Go Register (RNGEUG)

The RNG EU-Go is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

	0	63
Field	RNG EU-Go	
Reset	0	
R/W	W	
Addr	RNG 0x3_A050	

Figure 17-48. RNG EU-Go Register (RNGEUG)

17.4.5.8 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 the RNG output FIFO. Underflows caused by reading or writing the RNG output FIFO are reflected in RNGISR. Also, a write to the RNG output FIFO space will be reflected as an addressing error in RNGISR.

17.4.6 Advanced Encryption Standard Execution Unit (AESU)

This section contains details about the advanced encryption standard execution unit (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.

17.4.6.1 AESU Mode Register (AESUMR)

The AESU mode register, shown in Figure 17-49, contains 7 bits which are used to program the AESU. 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.

AESUMR is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If AESUMR is modified during processing, a context error will be generated.

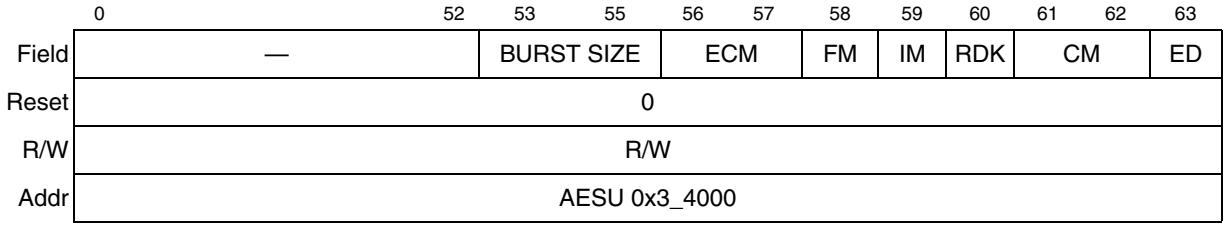


Figure 17-49. AESU Mode Register (AESUMR)

Table 17-15 describes AESUMR fields.

Table 17-38. AESUMR Field Descriptions

Bits	Name	Description
0–52	—	Reserved
53–55	BURST SIZE	Implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/context. The AESU signals to the channel that a BURST SIZE amount of data is available to be pushed to the FIFO. The inclusion of this field in the AESUMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the AESU.
56–57	ECM	Extend cipher mode: Used in combination with CM to define the mode of AES operation. See Table 17-39 for mode bit combinations.
58	FM	Final MAC: Processes final message block and generates final MAC tag at end of message processing (CCM mode only) 0 Do not generate final MAC tag. 1 Generate final MAC tag after CCM processing is complete.
59	IM	Initialize MAC: Initializes AESU for new message (CCM mode only) 0 Do not initialize (context will be loaded by host). 1 Initialize new message with nonce.

Table 17-38. AESUMR Field Descriptions (continued)

Bits	Name	Description
60	RDK	Restore decrypt key: Specifies that key data write will contain pre-expanded key (decrypt mode only). See note on use of RDK bit. 0 Expand the user key prior to decrypting the first block 1 Do not expand the key. The expanded decryption key will be written following the context switch.
61–62	CM	Cipher mode: Used in combination with ECM to define the mode of AESU operation. See Table 17-39 for mode bit combinations.
63	ED	Encrypt/Decrypt. If set, AESU operates the encryption algorithm; if not set, AESU operates the decryption algorithm. 0 Perform decryption. 1 Perform encryption. **Note:** This bit is ignored if CM is set to '11'–CTR Mode.

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 Figure 17-57 for more information on how SRT mode reduces context loading overhead.

Table 17-39. AES Cipher Modes

Mode	ECM	CM
ECB	00	00
CBC	00	01
Res	xx	10
CTR	00	11
SRT[1]	01	11
CCM	10	00

[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 17.4.6.9.3, "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 17-6 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).

17.4.6.2 AESU Key Size Register (AESUKSR)

The AESU key size register stores the number of bytes in the key (16,24,32). Any key data beyond the number of bytes in the key size register will be ignored. This register is cleared when the AESU is reset or re-initialized. If a key size other than 16, 24, or 32 bytes is specified, an illegal key size error will be generated. If AESUKSR 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.

	0 — 51	52 — 63
Field	—	Key Size
Reset	0	
R/W	R/W	
Addr	AESU 0x3_4008	

Figure 17-50. AESU Key Size Register (AESUKSR)

17.4.6.3 AESU Data Size Register (AESUDSR)

This AESU data size register is used to advise the AESU of the size of the data to be processed. Depending on the AES mode selected, data size must be divisible by a specific block size or a data size error will occur. In ECB, CBC, and CTR mode, the AESU does not automatically pad messages out to 128-bit blocks; therefore, when operating in ECB, CBC, or CTR mode, the message processed by the AESU must be divisible by 128 bits or a data size error will occur. When operating in CCM mode, data size must be divisible by 8-bits or a data size error will occur.

In normal operation, the full message length to be encrypted or decrypted with the AESU is copied from the descriptor to the AESUDSR; however only bits 56–63 are checked to determine if there is a data size error. If bits 57–63 are all zeros, the message is evenly divisible into 128-bit blocks.

This register is cleared when the AESU is reset or re-initialized. If a data size other than 128 bits is specified, an illegal data size error will be generated. Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of Data Size is modified during processing, a context error 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.

	0 — 51	52 — 63
Field	—	Data Size
Reset	0	
R/W	R/W	
Addr	AESU 0x3_4010	

Figure 17-51. AESU Data Size Register (AESUDSR)

17.4.6.4 AESU Reset Control Register (AESURCR)

This register allows three levels of reset of just the AESU, as defined by the three self-clearing bits:

	0 — 60	61	62	63
Field	—	RI	MI	SR
Reset	0			
R/W	R/W			
Addr	AESU 0x3_4018			

Figure 17-52. AESU Reset Control Register (AESURCR)

Table 17-17 describes AESU reset control register fields.

Table 17-40. AESURCR Field Descriptions

Bits	Names	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes AESU interrupts signaling DONE and ERROR to be reset. It further resets the state of the AESU interrupt status register. 0 Do not reset. 1 Reset interrupt logic.
62	MI	Module initialization. MI 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 Do not reset. 1 Reset most of AESU.
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the 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 Do not reset. 1 Full AESU reset.

17.4.6.5 AESU Status Register (AESUSR)

The AESU status register is a read-only register that reflects the state of six status outputs. Writing to AESUSR will result in an address error being reflected in the AESUISR.

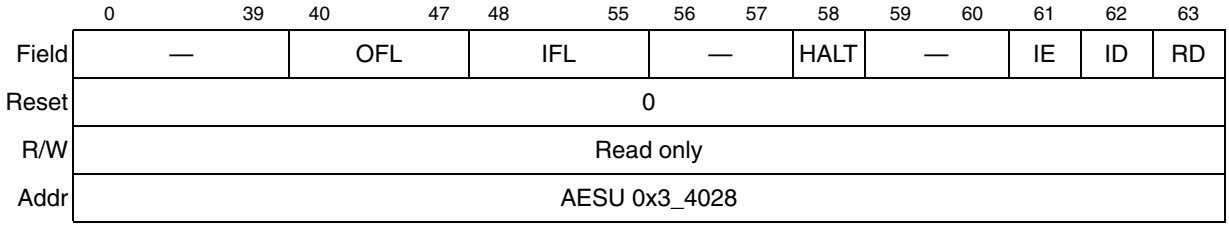


Figure 17-53. AESU Status Register (AESUSR)

Table 17-23 describes AESUSR fields.

Table 17-41. AESUSR Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved

Table 17-41. AESUSR Field Descriptions (continued)

Bits	Name	Description
58	HALT	Indicates that the AESU has halted due to an error. 0 AESU not halted 1 AESU halted **Note:** Because the error causing the AESU 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.
59–60	—	Reserved
61	IE	Interrupt error.This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 AESU is not signaling error 1 AESU is signaling error
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 17.6.2.2, “Interrupt Status Register (ISR)”). 0 AESU is not signaling done 1 AESU is signaling done
63	RD	Reset done. This status bit, when high, indicates that AESU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

17.4.6.6 AESU Interrupt Status Register (AESUISR)

The AESU interrupt status register, Figure 17-54, tracks the state of possible errors, if those errors are not masked, through the AESUICR.

	0–50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	0													
R/W	Read only													
Addr	AESU 0x3_4030													

Figure 17-54. AESU Interrupt Status Register (AESUISR)

Table 17-19 describes AESU interrupt register fields.

Table 17-42. AESUISR Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while the AESU was processing. 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 AESU.

Table 17-42. AESUISR Field Descriptions (continued)

Bits	Name	Description
52	ERE	Early read error. The AESU IV register was read while the AESU was processing. 0 No error detected 1 Early read error
53	CE	Context error. An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while AESU was processing 0 No error detected 1 Context error
54	KSE	Key size error. An inappropriate value (not 16, 24 or 32 bytes) was written to the AESU Key Size Register 0 No error detected 1 Key size error
55	DSE	Data size error. A value was written to the AESU Data Size Register that is not a multiple of 128 bits. 0 No error detected 1 Data size error
56	ME	Mode error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Valid Data 1 Reserved or invalid mode selected
57	AE	Address error. An illegal read or write address was detected within the AESU address space. 0 No error detected 1 Address error
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 AESU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the AESU 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

17.4.6.7 AESU Interrupt Control Register (AESUICR)

The AESU interrupt control register, shown in Figure 17-55, controls the result of detected errors. For a given error, as defined in Section 17.4.6.6, “AESU Interrupt Status Register (AESUISR),” if the corresponding bit in this register is set, the error is ignored; no error interrupt occurs and AESUISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, AESUISR

is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

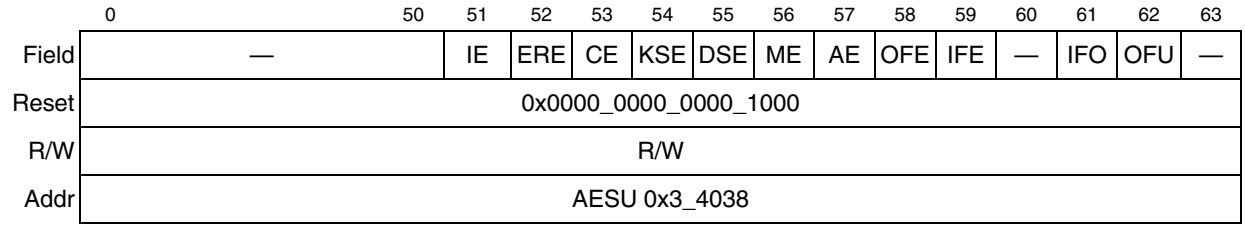


Figure 17-55. AESU Interrupt Control Register (AESUICR)

Table 17-43 describes the fields of the AESUICR.

Table 17-43. AESUICR Field Descriptions

Bits	Name	Description
0–50	—	Reserved
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. One of AESUKR, the AESUKSR, AESUDSR, AESUMR, or AESU 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

Table 17-43. AESUICR Field Descriptions (continued)

Bits	Name	Description
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

17.4.6.8 AESU End of Message Register (AESUEMR)

The AESU end of message register, shown in Figure 17-56, is used to indicate an AES operation may be completed. After the final message block is written to the input FIFO, AESUEMR must be written. The value in AESUDSR will be used to determine how many bits of the final message block (always 128) will be processed. Writing to AESUEMR causes the AESU to process the final block of a message, allowing it to signal DONE. A read of this register will always return a zero value. AESUEMR is only used when the is operated as a slave. The descriptors and crypto-channel activate the AESU (through an internally generated write to the end of message register) when the SEC acts as an initiator.

	0	63
Field	AESU End of Message	
Reset	0	
R/W	W	
Addr	AESU 0x3_4050	

Figure 17-56. AESU End of Message Register

17.4.6.9 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 17-57.

Cipher Mode	Context Register (64 Bits Each) 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 17-57. AESU Context Registers

17.4.6.9.1 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 least significant bytes of the initialization vector (bytes 1–8).
- IV2 holds the most significant bytes of the initialization vector (bytes 9–16).

The IV must be written prior to the message data. If the IV registers are written during message processing, or the 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 17.4.6.5, “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).

17.4.6.9.2 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 XORed 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 17-44

Table 17-44. Counter Modulus

Value Written	Modulus
8	2^8
16	2^{16}
24	2^{24}
32	2^{32}
40	2^{40}
48	2^{48}
56	2^{56}
64	2^{64}
72	2^{72}
80	2^{80}
88	2^{88}
96	2^{96}
104	2^{104}
112	2^{112}
120	2^{120}
128	2^{128}

17.4.6.9.3 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 17-44.

The only difference between SRT mode and CTR mode is in SRT mode, the AES context is loaded and read through 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.

17.4.6.9.4 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 is 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 follows.

The context for CCM encryption/MAC generation is:

- Reg 1–2 Session specific128 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 the 802.11i standard 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 128bits 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 standard frame. Note that 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 have been output, the CCM encrypt operation is complete.

The context for CCM decryption/MAC generation is:

- Reg 1–2 Session specific128 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 the 802.11i standard 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

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 standard 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 standard 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 standard 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.

17.4.6.9.5 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.

17.4.6.9.6 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.

17.5 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 17.3.1, "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.
- Upon notification of completion of the EU reset sequence, initialize mode registers in the assigned EU.
- Initialize EUs and write to EU registers such as key size and text-data size.
- Transfer data parcels (up to 32 Kbytes) from system memory into assigned EU input registers and FIFOs. This may involve using link tables to gather input data that has been split into multiple segments which are stored in various locations of system memory.
- Transfer data parcels (up to 32 Kbytes) from assigned EU output registers and FIFOs to system memory space. This may involve using link tables to scatter output data into multiple segments which are stored in various locations of system memory.

- Initialize the EU-Go register (where applicable) in the assigned EU upon completion of last EU write indicated by the descriptor. The channel will wait for a indication from the EU that processing of input text-data is complete before proceeding with further activity after writing EU-Go.
- Reset assigned EU(s).
- Release assigned EU(s).
- When a descriptor has been completely processed, provide feedback to the host, in the form of interrupt and/or descriptor header write-back to system memory.
- When descriptor processing is halted due to an error, provide feedback to the host through interrupt.

The channel will wait indefinitely for the controller to complete a requested activity before continuing to the next step of descriptor processing.

17.5.1 Crypto-Channel Registers

Crypto-channel registers are described in the following sections.

17.5.1.1 Crypto-Channel Configuration Register (CCCR)

This register contains five operational bits permitting configuration of the crypto-channel as shown in Figure 17-58. Table 17-45 describes the CCCR.

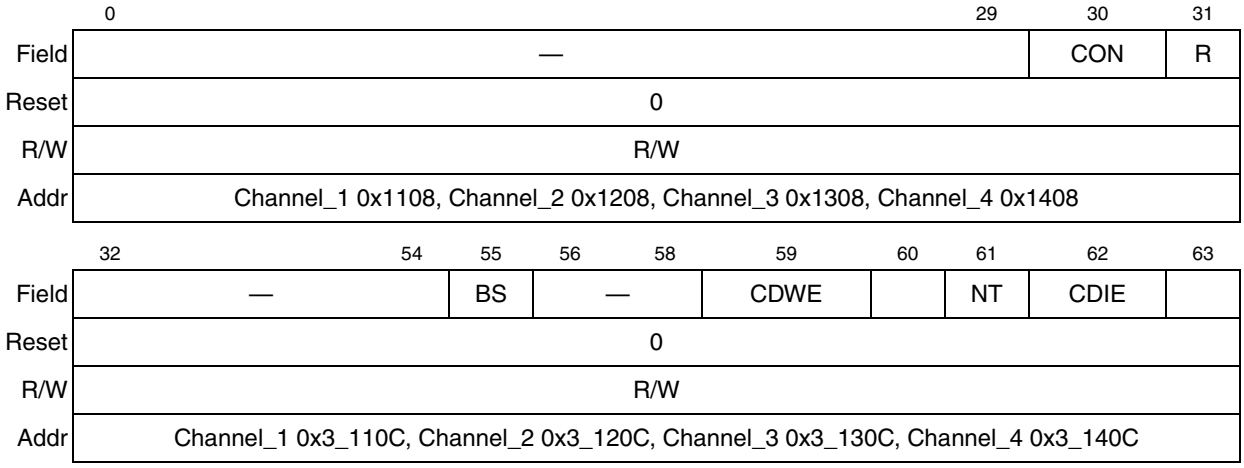


Figure 17-58. Crypto-Channel Configuration Register (CCCR)

Table 17-45. 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.

Table 17-45. CCCR Field Descriptions (continued)

Bits	Name	Description
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
56-58	—	Reserved, set to zero
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, then notify the host by writing back the descriptor header with 0xFF in bits 0-7. 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. Channel DONE notification type. This bit controls when the crypto-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 WE control bits. 0 Global: The crypto-channel will generate channel done notification (if enabled) at the end of each descriptor. 1 Done bit: The crypto-channel will generate channel done notification (if enabled) at the end of every descriptor with the Done bit set in the descriptor header.
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 17.5.2, “Interrupts,” for complete description of channel interrupt operation.
63	—	Reserved, set to zero

17.5.1.2 Crypto-Channel Pointer Status Register (CCPSR)

This register contains status fields and counters which provide the user with status information regarding the channel's actual processing of a given descriptor.

	0–2	3–7	8–11	12–15	16–19	20–23	24–31
Field	—	FF_COUNTER	—	G_STATE	—	S_STATE	CHN_STATE
Reset	0						
R/W	Read only						
Addr	Channel_1 0x3_01110, Channel_2 0x3_01210, Channel_3 0x3_01310, Channel_4 0x3_01410						

	32–37	38	39	40	41	42	43	44	45	46	47	48–59	60–63
Field		MI	MO	PR	SR	PG	SG	PRD	SRD	PD	SD	Error	PAIR_PTR
Reset	0x0000 _0007												
R/W	Read only												
Addr	Channel_1 0x3_01114, Channel_2 0x3_01214, Channel_3 0x3_01314, Channel_4 0x3_01414												

Figure 17-59. Crypto-Channel Pointer Status Register (CCPSR)

Table 17-46 describes the crypto-channel pointer status register fields.

Table 17-46. CCPSR Field Descriptions

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. Reflects the state of the crypto-channel gather control state machine. The value of this field indicates which stage the crypto-channel is in while performing gather function. Table 17-47 lists all possible values of the G_STATE field. **Note:** G_STATE is documented for information only. The user will not typically care about the gather state machine.
16–19	—	Reserved.
20–23	S_STATE	Scatter state machine state. Reflects the state of the crypto-channel scatter control state machine. The value of this field indicates which stage the crypto-channel is while performing scatter function. Table 17-48 lists all possible values of the S_STATE field. **Note:** S_STATE is documented for information only. The user will not typically care about the scatter state machine.
24–32	CHN_STATE	Crypto-channel state machine state. Reflects the state of the crypto-channel control state machine. The value of this field indicates exactly which stage the crypto-channel is in the sequence of fetching and processing data descriptors. Table 17-49 lists all possible values of the CHN_STATE field. **Note:** CHN_STATE is documented for information only. The user will not typically care about the crypto-channel state machine.
31–37	—	Reserved, set to zero

Table 17-46. CCPSR Field Descriptions (continued)

Bits	Name	Description
38	MI	Multi_EU_IN. The Multi_EU_IN bit 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
39	MO	Multi_EU_OUT. The Multi_EU_OUT bit 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	PRI_REQ. Request primary EU assignment. 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 OP_0 field in the descriptor header register as long as this bit remains set. 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.
41	SR	SEC_REQ. Request secondary EU assignment. 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. 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.
42	PG	Primary EU granted. 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. 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. 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. 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 17-46. CCPSR Field Descriptions (continued)

Bits	Name	Description
47	SD	Secondary EU done. 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. 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 17-50 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 17-51 shows the meaning of all possible values of the PAIR_PTR field.

Table 17-49 shows the values of the gather state machine.

Table 17-47. G_STATE Field Values

Value	Gather State Machine
0x0	idle
0x1	load_4pointers_frm_gather_table
0x2	load_4pointers_frm_gather_table_done
0x3	next_bit_set_load_next_gather_table
0x4	process_gather_pointer
0x5	request_block_data_trans
0x6	request_block_data_trans_done
0x7	request_bytes_data_trans
0x8	request_bytes_data_trans_done
0x9	increase_table_pointer
0xA	update_gather_pointer
0xB	gather_table_done
0xC	gather_error
0xD	load_next_4pointers_frm_gather_table
0xE-F	reserved

Table 17-49 shows the values of the scatter state machine.

Table 17-48. 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_trans
0x8	request_bytes_data_trans_done
0x9	increase_table_pointer
0xA	update_scatter_pointer
0xB	scatter_table_done
0xC	scatter_error
0xD	load_next_4pointers_frm_scatter_table
0xE-F	reserved

Table 17-49 shows the values of crypto-channel states.

Table 17-49. CHN_STATE Field Values

Value	Crypto-Channel State
0x00	Idle
0x01	Process_header
0x02	Fetch_descriptor
0x03	Channel_done
0x04	Channel_done_irq
0x05	Channel_done_writeback
0x06	Channel_done_notification
0x07	Channel_error
0x08	Request_pri_eu
0x09	Inc_data_pair_pointer
0x0A	Delay_data_pair_update
0x0B	Evaluate_data_pairs
0x0C	Write_reset_pri
0x0D	Release_pri_eu
0x0E	Write_reset_sec

Table 17-49. CHN_STATE Field Values (continued)

Value	Crypto-Channel State
0x0F	Release_sec_eu
0x10	Process_data_pairs
0x11	Write_mode_pri
0x12	Write_mode_sec
0x13	Write_datasize_pri
0x14	Delay_rng_done
0x15	Write_datasize_sec_multi_eu_in
0x16	Trans_request_read_multi_eu_in
0x17	Delay_pri_sec_done
0x18	Trans_request_read
0x19	Write_key_size
0x1A	Write_EU-Go
0x1B	Delay_pri_done
0x1C	Write_reset_irq_pri
0x1D	Write_reset_irq_sec
0x1E	Write_datasize_sec_snoopout
0x1F	Trans_request_write_snoopout
0x20	Delay_sec_done
0x21	Trans_request_write
0x22	Evaluate_reset
0x23	Reset_write_reset_pri
0x24	Reset_release_pri_eu
0x25	Reset_write_reset_sec
0x26	Reset_release_sec_eu
0x27	Reset_channel
0x28	Write_datasize_pri_post
0x29	Reset_release_all
0x2A	Reset_release_all_delay
0x2B	Request_sec_eu
0x2C	Write_datasize_sec
0x2D	Write_pri_EU-Go_multi_eu_out
0x2E	Write_sec_EU-Go_multi_eu_out
0x2F	Write_pri_EU-Go_multi_eu_in
0x30	Write_sec_EU-Go_multi_eu_in
0x31	Write_datasize_pri_delay
0x32	Wait_AFEU_done
0x33	Trans_extend_write

Table 17-49. CHN_STATE Field Values (continued)

Value	Crypto-Channel State
0x34	Trans_extend_a
0x35	Trans_extend_b
0x36- 0xFF	Reserved

Table 17-50 shows the bit positions of each potential error. Multiple errors are possible.

Table 17-50. CCPSR 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 through the CCCR register. This bit can be cleared by writing ‘1’ to this bit in the CCPSR register.
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 bit in the CCPSR register.
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: • Invalid primary EU indicated by OP_0 field in descriptor header. • Invalid secondary EU indicated by OP_1 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 17-51 shows the possible values of the PTR_DW field in the CCPSR.

Table 17-51. 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
0x05	Processing pointer dword 5
0x06	Processing pointer dword 6
0x07	Complete (or not yet begun) processing of header dword and pointer dwords
0x08-FF	Reserved

17.5.1.3 Crypto-Channel Current Descriptor Pointer Register (CDPR)

The CDPR, shown in Figure 17-60, contains the address of the descriptor which the crypto-channel is currently processing. This register, 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.

	0 — 31	32 — 63
Field	—	CUR_DES_PTR_ADRS
Reset	0x0000_0000	
R/W	Read only	
Addr	Channel_1 0x3_1140, Channel_2 0x3_1240, Channel_3 0x3_1340, Channel_4 0x3_1440	

Figure 17-60. Crypto-Channel Current Descriptor Pointer Register (CDPR)

The fields in CDPR perform the functions described in Table 17-52.

Table 17-52. CDPR Field Descriptions

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. 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.

17.5.1.4 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, displayed in Figure 17-61, 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 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.

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.

	0 — 31	32 — 63
Field	—	FETCH_ADRS
Reset	0x0000_0000	
R/W	W	
Addr	Channel_1 0x3_01148, Channel_2 0x3_01248, Channel_3 0x3_01348, Channel_4 0x3_01448	

Figure 17-61. Fetch FIFO

Table 17-53 describes the fetch FIFO fields.

Table 17-53. 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.

17.5.1.5 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 17.3.1, "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 17-62. Data Packet Descriptor Buffer

17.5.2 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 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 through its interrupt mask register (IMR). See Section 17.6.2.1, “Interrupt Mask Register (IMR).”

17.5.2.1 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 17-58). Assuming 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.

17.5.2.2 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 the ERROR field of the channel pointer status register (CPSR). Refer to Table 17-50 for a complete listing of error types.

17.5.2.3 Channel Reset

Channel reset is asserted when the host sets the R (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 R 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 R bit and return the control state machine to the idle state.
- If the R 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 R bit and return the control state machine to the idle state.

17.6 SEC Controller

The SEC controller 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 through the master/slave bus interface and to the channels and EUs through 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

17.6.1 Controller Registers

The controller registers are described in detail in the following sections.

17.6.2 EU Assignment Status Register (EUASR)

The EUASR, displayed in Figure 17-63, 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 17-54) indicates the channel to which an EU is assigned.

	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
Field	—	AFEU	—	MDEU	—	AESU	—	DEU
Reset	0xF	0x0	0xF	0x0	0xF	0x0	0xF	0x0
R/W	Read only							
Addr	0x3_1028							

	32–35	36–39	40–43	44–47	48–51	52–55	56–59	60–63
Field	—				—	PKEU	—	RNG
Reset	0x00FF				0xF	0x0	0xF	0x0
R/W	Read only							
Addr	0x3_102C							

Figure 17-63. EU Assignment Status Register (EUASR)

Table 17-54 shows the EU channel assignments.

Table 17-54. 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

17.6.2.1 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 17-64 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 17-55.

	0 — 14	15
Field	—	ITO
Definition		
Reset	0x0000	
R/W	R/W	
Addr	0x3_1008	

	16 — 19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	DONE Overflow				CHN_4		CHN_3		CHN_2		CHN_1	
Definition		4	3	2	1	Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000												
R/W	R/W												
Addr	0x3_1008												

	32 — 39	40	41	42	43	44	45	46	47
Field		—		PKEU		—		RNG	—
Definition				Err	Dn			Err	
Reset	0x0000								
R/W	R/W								
Addr	0x3_100C								

	48	49	50	51	52	53	54	55	56	57	58 — 63	
Field	—		AFEU		—		MDEU		—		AESU	—	DEU	
Definition			Err	Dn			Err	Dn			Err	Dn	Err	Dn
Reset	0x0000													
R/W	R/W													
Addr	0x3_100C													

Figure 17-64. Interrupt Mask Register (IMR)

17.6.2.2 Interrupt Status Register (ISR)

The ISR contains fields representing all possible sources of interrupts. The interrupt status register is cleared either by a reset, or by writing the appropriate bits active in the interrupt clear register. Figure 17-65 shows the bit positions of each potential interrupt source. The bit fields are described in Table 17-55.

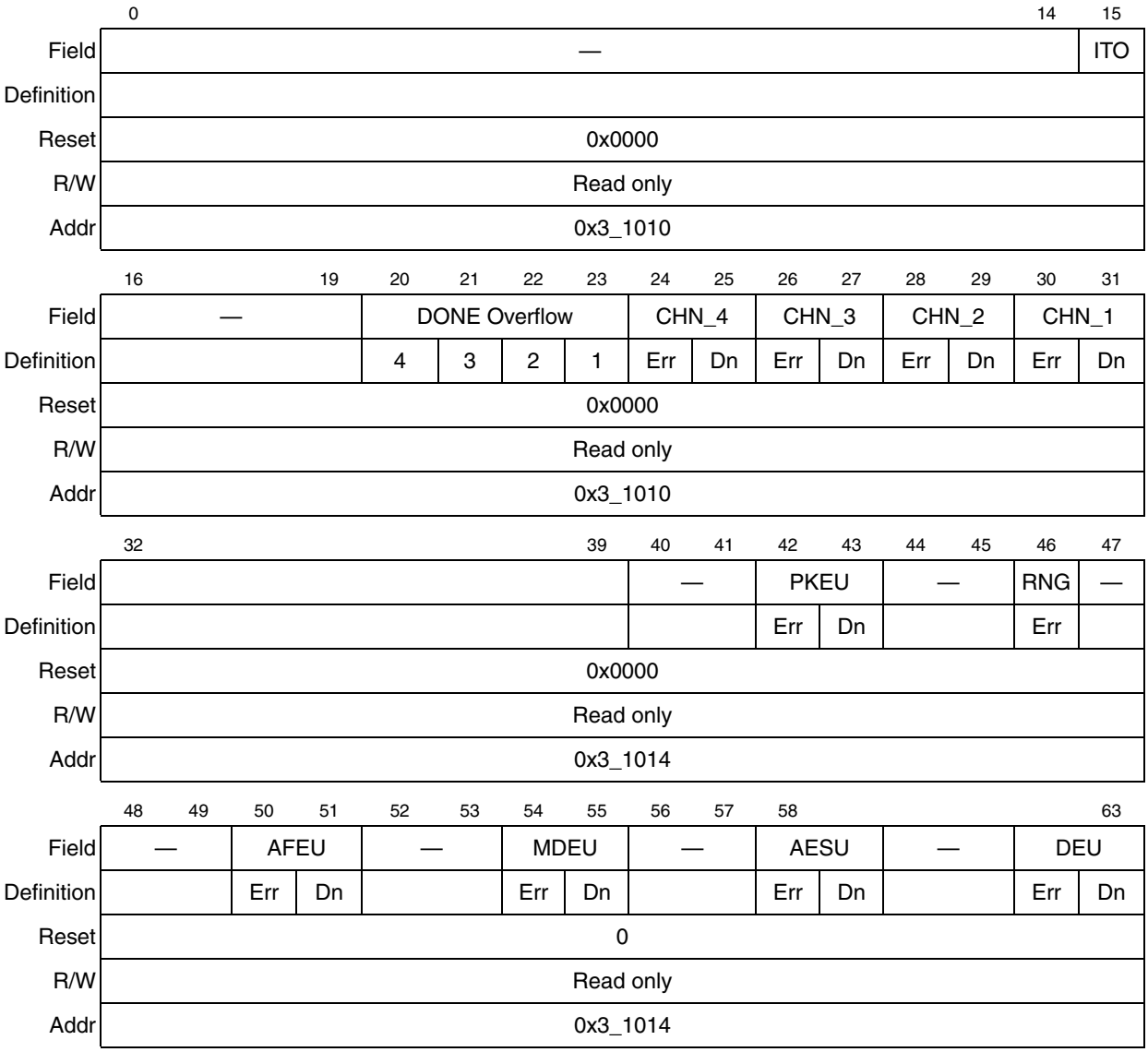


Figure 17-65. Interrupt Status Register (ISR)

17.6.2.3 Interrupt Clear Register (ICR)

The interrupt control register 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 $\overline{\text{IRQ}}$ (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 $\overline{\text{IRQ}}$, will be reasserted shortly thereafter. Figure 17-66 shows the bit positions of each interrupt source that can be cleared by this register. The complete bit definitions for the ICR can be found in Figure 17-66. The bit fields are described in Table 17-55.

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.

	0 — 14	15
Field	—	ITO
Definition		
Reset	0x0000	
R/W	W	
Addr	0x3_1018	

	16 — 19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	DONE Overflow				CHN_4		CHN_3		CHN_2		CHN_1	
Definition		4	3	2	1	Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000												
R/W	W												
Addr	0x3_1018												

	32 — 39	40	41	42	43	44	45	46	47
Field	—	—		PKEU		—		RNG	—
Definition				Err	Dn			Err	
Reset	0x0000								
R/W	W								
Addr	0x3_101C								

	48 — 49	50	51	52	53	54	55	56	57	58 — 63
Field	—	AFEU		—		MDEU		—		AESU	—	DEU	
Definition		Err	Dn			Err	Dn			Err	Dn	Err	Dn
Reset	0x0000												
R/W	W												
Addr	0x3_101C												

Figure 17-66. Interrupt Clear Register (ICR)

Table 17-55 describes the interrupt mask, status, and clear register fields.

Table 17-55. Interrupt Mask, Status, and Clear Register Field Descriptions

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 and 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.
Multiple	EU_Err_Dn	Each of the execution units has error and 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.
0–14, 16–19, 32–41, 44–45, 47–49, 52–53, 56–57, 60–61	—	Reserved, set to zero

17.6.2.4 ID Register

The read-only ID register, displayed in Figure 17-67, contains a 64-bit value that uniquely identifies the version of the SEC 2.0. The value of this register is always 0x0000_0000_0000_0040, indicating that this is the first version of the SEC 2.0.

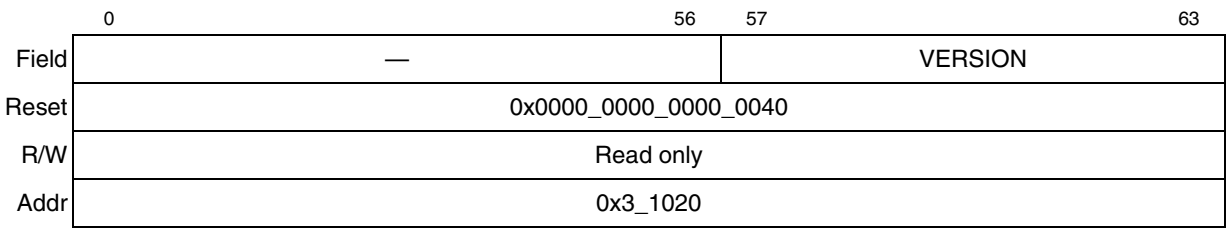


Figure 17-67. ID Register

17.6.2.5 Master Control Register (MCR)

The MCR, shown in Figure 17-68, controls certain functions in the controller and provides a means for software to reset the SEC.

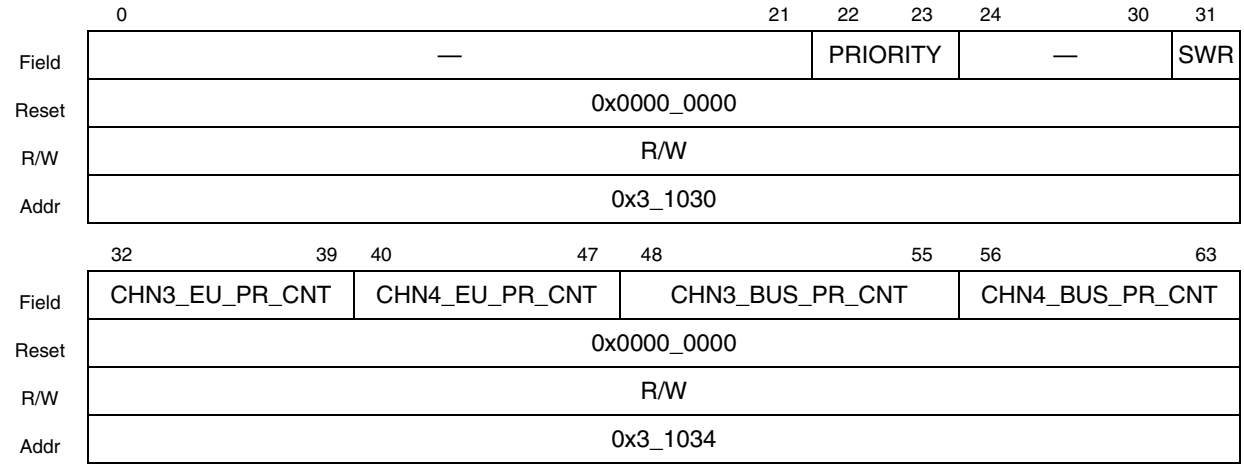


Figure 17-68. Master Control Register (MCR)

Table 17-56 describes the master control register signals.

Table 17-56. MCR Field Descriptions

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 MPC8555E 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 real time. 00 Lowest priority (default) 01 Next lowest priority 10 Next highest priority 11 Highest priority
24–30	—	Reserved
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 **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 purely round robin basis. If set to nonzero, CHN4_EU_PR_CTR must also be set to a different, nonzero value.

Table 17-56. MCR Field Descriptions (continued)

Bits	Name	Description
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 purely round robin basis. If set to nonzero, CHN3_EU_PR_CTR must also be set to a different, nonzero 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 purely round robin basis. If set to nonzero, CHN4_BUS_PR_CTR must also be set to a different, nonzero 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 purely round robin basis. If set to nonzero, CHN3_BUS_PR_CTR must also be set to a different, nonzero value.

17.6.2.6 EU Access

Assignment of an EU function to a channel is done dynamically. With dynamic assignment, the channel requests an 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.

17.6.2.7 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 17.7.3, “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.

17.6.2.8 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 on a 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 nonzero value, the arbiter will implement the weighted priority scheme. Otherwise, the arbitration will be round-robin. Setting only one of the CHN*n*_EU_PR_CNT fields to a nonzero value will result in unpredictable operation.

17.6.2.9 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 expires

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; that is, EU access may be prioritized, while bus access is purely round robin, and vice-versa.

17.6.2.10 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, the arbiter satisfies those requests through a round-robin scheme as the resource becomes available. When all requests have been satisfied, the arbiter takes another snapshot.

17.7 Bus Interface

The controller in the SEC (refer to Section 17.6, “SEC 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.

17.7.1 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 through 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 17.6.2.9, "Priority Arbitration," for more information.

17.7.1.1 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.

17.7.1.1.1 Slave Aborts

It is possible for the intended slave of an SEC master-initiated transaction to terminate the transfer due to an error. The SEC's transaction requests are posted to the MPC8555E target queue, after which the MPC8555E takes responsibility for completing the transaction or signaling error. An error in an SEC-initiated transaction will also be reported by the SEC through the channel interrupt status register. The host will be able to determine which channel generated the interrupt by checking the ISR for the channel ERROR bit.

17.7.1.2 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 and 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; that is, all bytes have been transferred.

17.7.1.2.1 Slave Access

The controller also acts as a bus slave. As a slave, the controller simply responds to read and write commands from the bus. When a write command is received from the bus, the controller takes the data from the slave interface and sends it to whichever internal location is 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.

17.7.2 Bus Arbitration Priority

Transaction priority is configured for all channels in the SEC master control register (MCR[Priority]) as seen in Figure 17-68. The SEC does not dynamically adjust its transaction priority; however, system software can adjust SEC transaction priority in real time, with the change in priority taking effect immediately.

17.7.3 Snooping by Caches

All SEC transactions are snooped by the MPC8555E coherency module. This is part of the wiring of the SEC interface and requires no user intervention.

17.7.4 Interrupts

The SEC generates a single interrupt to the MPC8555E programmable interrupt controller. (Refer to Section 10.1.5.2, “Internal Interrupt Sources,” for additional information.) The user allows interrupts from the SEC to be reported to the CPU by clearing the mask bit in the associated vector/priority register of the PIC.

The user can control which events cause an interrupt by configuring the SEC interrupt mask register. These events are:

- 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 source of that interrupt. To clear an interrupt, the user should first write 1 to the bits in the SEC interrupt clear register (ICR) corresponding to the pending ISR bits, and then write zeros to the end of interrupt register in the PIC to enable additional SEC interrupts.

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 which caused the interrupt will cause the interrupt to be asserted again a few cycles later.

17.8 Power-Saving Mode

The SEC may be disabled by setting DEVDISR[SEC]. (Refer to Section 18.4.1.11, “Device Disable Register (DEVDISR).”) The clocks to the SEC are active by default. The SEC should not be enabled/disabled during normal operation.

Part IV
Global Functions and Debug

This part defines other global blocks of the MPC8555E. The following chapters are included:

- Chapter 18, “Global Utilities,” defines the global utilities of the MPC8555E. These include power management, I/O device enabling, power-on reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals
- Chapter 19, “Performance Monitor,” describes the performance monitor of the MPC8555E.
- Chapter 20, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8555E.

Chapter 18
Global Utilities

This chapter describes the global utilities of the MPC8555E. It provides signal descriptions, register descriptions, and a functional description of these utilities.

18.1 Overview

The global utilities block controls power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal configuration, alternate function selection for multiplexed signals, and clock control.

18.2 Global Utilities Features

This section provides an overview of global utilities features.

18.2.1 Power Management and Block Disables

The following features affect the device's overall power consumption:

- Dynamic power management mode
- Software-controlled power management (doze, nap, sleep)
- Externally controlled power management (doze, sleep)
- Static power management (I/O block disables)

18.2.2 Accessing Current POR Configuration Settings

The POR configuration values of all device parameters sampled from pins at reset are available through memory-mapped registers in the global utilities block.

18.2.3 General-Purpose I/O

The PCI and TSEC2 data bus signals can be used as general-purpose I/O signals when not used for their primary function. Memory-mapped registers in the global utilities block provide control and status for the use of these signals. A general purpose input register is loaded with the values of the local bus address/data pins at the negation of $\overline{\text{HRESET}}$.

18.2.4 Interrupt and Local Bus Signal Multiplexing

IRQ[9:11] and $\overline{\text{LCS}}$[5:7] serve multiple functions that can be selected by configuration registers in the global utilities block.

The multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 45, "Parallel I/O Ports," for details on CPM signal multiplexing.

18.2.5 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

18.3 External Signal Description

The following sections provide information about signals that serve as global utilities.

18.3.1 Signals Overview

Table 18-1 summarizes the external signals used by the global utilities block.

Table 18-1. External Signal Summary

Signal Name	I/O	Description	Reference (Section/Page)
ASLEEP	O	Signals that the device has reached a sleep state.	18.5.1.5.3/18-24
$\overline{\text{CKSTP\_IN}}$	I	Checkstop input	—
$\overline{\text{CKSTP\_OUT}}$	O	Checkstop output.	—
CLK_OUT	O	Clock out. Selected by CLKOCR values.	18.4.1.16/18-19

18.3.2 Detailed Signal Descriptions

Table 18-2 describes signals in the global utilities block in detail.

Table 18-2. Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP	O	Asleep. See Section 18.5.1.5.3, "Sleep Mode." After negation of $\overline{\text{HRESET}}$, ASLEEP is asserted until the device completes its power-on reset sequence and reaches its ready state.	
		State Meaning	Asserted—Indicates that the device is either still in its power-on reset sequence or it has reached a sleep state after a power-down command is issued by software. Negated—The device is not in sleep mode. (It has either awakened from a power-down state, or has completed the POR sequence.)
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Negates synchronously with SYSCLK when leaving power-on sequence; otherwise negation is asynchronous.
$\overline{\text{CKSTP\_IN}}$	I	Checkstop in	
		State Meaning	Asserted—Indicates that the e500 core must enter a hard stop condition. All e500 clocks are turned off. $\overline{\text{CKSTP\_OUT}}$ is asserted. The rest of MPC8555E device logic, including memory controllers, internal memories and registers, and I/O interfaces, remains functional. Negated—Indicates that normal operation should proceed.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the MPC8555E is reset with assertion of $\overline{\text{HRESET}}$.

Table 18-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{CKSTP\_OUT}}$	O	Checkstop out	
		State Meaning	Asserted—Indicates that the e500 core of the MPC8555E is in a checkstop state. The rest of the MPC8555E logic remains functional. Negated—Indicates normal operation. After $\overline{\text{CKSTP\_OUT}}$ has been asserted, it is negated after the next negation (low-to-high transition) of $\overline{\text{HRESET}}$.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the device has been reset with a hard reset.
CLK_OUT	O	Clock out. Reflects clock signal selected by CLKOCR (see Section 18.4.1.16, "Clock Out Control Register (CLKOCR)").	
		State Meaning	Asserted—If CLKOCR[ENB] = 1, clock signal selected by CLKOCR[CLK_SEL] is driven. High impedance—If CLKOCR[ENB] = 0.
		Timing	Assertion/Negation—Depends on the value of CLKOCR[CLK_SEL].

18.4 Memory Map/Register Definition

Table 18-3 summarizes the global utilities registers and their addresses. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 18-3. Global Utilities Block Register Summary

Offset	Register	Access	Reset	Section/Page
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_*n*1*nn*	18.4.1.1/18-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	18.4.1.2/18-5
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	R/W	0x000*n*_007F	18.4.1.3/18-6
0xE_000C	PORDEVSR—POR I/O device status register	R	see ref.	18.4.1.4/18-7
0xE_0010	PORDBGMSR—POR debug mode status register	R	see ref.	18.4.1.5/18-8
0xE_0020	GPPORCR—General-purpose POR configuration register	R	see ref.	18.4.1.6/18-9
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	18.4.1.7/18-10
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	18.4.1.8/18-11
0xE_0050	GPINDR—General-purpose input data register	R	0x*nnnn*_0000	18.4.1.9/18-12
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	18.4.1.9/18-12
Device Disables				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	18.4.1.11/18-14
Power Management Registers				
0xE_0080	POWMGTCSR—Power management status and control register	R/W	0x0000_0000	18.4.1.12/18-16

Table 18-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
Interrupt Reporting				
0xE_0090	MCPSUMR—Machine check summary register	Read/Clear	0x0000_0000	18.4.1.13/18-17
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	18.4.1.14/18-18
0xE_00A4	SVR—System version register	R	MPC8555E / MPC8541E system version	18.4.1.15/18-19
Debug Control				
0xE_0E00	CLKOCR—Clock out select register	R/W	0x0000_0000	18.4.1.16/18-19
0xE_0E20	LBDLLCR—LBC DLL control register	R/W	0x0000_0000	18.4.1.17/18-20

18.4.1 Register Descriptions

This section describes the global utilities registers in detail.

18.4.1.1 POR PLL Status Register (PORPLLSR)

PORPLLSR, shown in Figure 18-1, contains the settings for the PLL ratios as set by the cfg_sys_pll[0:3], cfg_core_pll[0:1], TSEC2_TXD[1], and TSEC2_TXD[0] POR configuration pins. See Section 4.4.3.1, “System PLL Ratio,” Section 4.4.3.10, “PCI Clock Selection,” and Section 4.4.3.2, “e500 Core PLL Ratio,” for more information.

	0								9	10	15	16	17					25	26	31
R	0	0	0	0	0	0	0	0	1	e500_Ratio		PCI1_clk_sel	PCI2_clk_sel	0	0	0	0	0	1	0	1	Plat_Ratio	0
W																							
Reset	0000_0000_01*nn_nnnn_nn*00_0001_01*nn_nnnn*																						
Offset	0xE_0000																						

Figure 18-1. POR PLL Status Register (PORPLLSR)

Table 18-4 describes the bit settings of PORPLLSR.

Table 18-4. PORPLLSR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–15	e500_Ratio	Clock ratio between the e500 core and the CCB clock. The 2 lsbs of this field correspond to the values on cfg_core_pll[0:1] at the negation of HRESET. Patterns not shown are reserved. 000100 2:1 000101 5:2 000110 3:1 000111 7:2
16	PCI1_clk_sel	Clock used for PCI1. This bit corresponds to the value on TSEC2_TXD[1] at the negation of HRESET: 0 PCI1 is clocked by PCI1_CLK 1 PCI1 is clocked by SYSCLK[1]
17	PCI2_clk_sel	Clock used for PCI2. This bit corresponds to the value on TSEC2_TXD[0] at the negation of HRESET: 0 PCI2 is clocked by PCI2_CLK 1 PCI2 is clocked by SYSCLK[1]
18–25	—	Reserved
26–30	Plat_Ratio	Clock ratio between the CCB (platform) clock and SYSCLK. The 4 lsbs correspond to the values on cfg_sys_pll[0:3] at the negation of HRESET. Patterns not shown are reserved. 00010 2:1[1] 01000 8:1 00011 3:1 01001 9:1 00100 4:1 01010 10:1 00101 5:1 01100 12:1 00110 6:1 10000 16:1
31	—	Reserved

[1] CCB to SYSCLK clock ratio of 2:1 must not be selected when TSEC2_TXD[1] or TSEC2_TXD[0] are configured to be clocked by SYSCLK.

18.4.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 18-2, reports setting of the POR configuration pins that control the boot mode settings (described in Section 4.4.3.3, “Boot ROM Location,” Section 4.4.3.5, “CPU Boot Configuration,” and Section 4.4.3.6, “Boot Sequencer Configuration,”) and the default settings of PCI host/agent mode (described in Section 4.4.3.4, “Host/Agent Configuration”).

	0	1			4	5	7	8	9	10	11	12		14	15	16																31
R	BCFG	0	0	0	0	ROM_LOC		0	0	BSCFG		0	0	0	PCI1_HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	*n*000_0*nn*n_00*nn*_000*n*_0000_0000_0000_0000																															
Offset	0xE_0004																															

Figure 18-2. POR Boot Mode Status Register (PORBMSR)

For more information about the PCI configurations, see Section 16.3.2.19, “PCI Bus Function Register (PBFR).” Figure 18-5 describes the bit settings of the PORBMSR.

Table 18-5. PORBMSR Field Descriptions

Bits	Name	Description
0	BCFG	CPU boot configuration 0 The CPU is prevented from booting until configuration by an external master is complete. 1 The CPU is allowed to start fetching boot code.
1–4	—	Reserved
5–7	ROM_LOC	Location of boot ROM 000 PCI1 001 DDR SDRAM 010 PCI2 011–100 Reserved 101 Local bus GPCM: 8-bit 110 Local bus GPCM:16-bit 111 Local bus GPCM: 32-bit
8–9	—	Reserved
10–11	BSCFG	Boot sequencer configuration 00 Reserved 01 Boot sequencer enabled with normal I^2C addressing 10 Boot sequencer enabled with extended I^2C addressing 11 Boot sequencer disabled
12–14	—	Reserved
15	PCI1_HA	PCI1 host/agent mode configuration. When the MPC8555E is an agent on an interface, it is prevented from mastering transactions on that interface until the external host configures the interface appropriately. Note that PCI2 is always in host mode. 0 PCI1 agent mode 1 PCI1 host mode
16–31	—	Reserved

18.4.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

PORIMPSCR, shown in Figure 18-3, defines the current I/O driver impedances for local bus signals and reports the I/O impedance setting for the PCI interface.

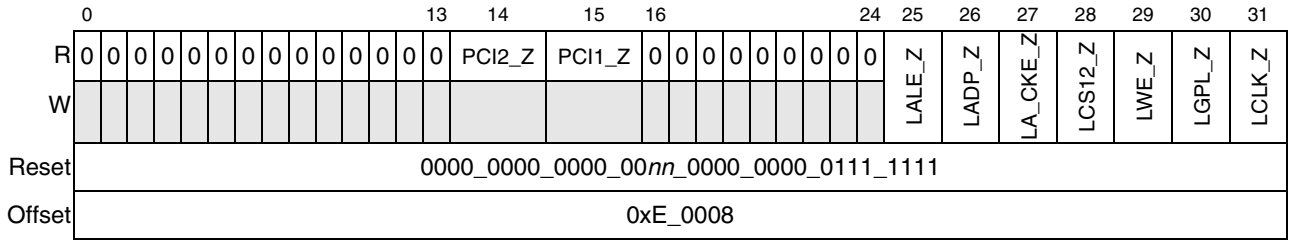


Figure 18-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

The I/O impedance of the local bus signal (including the local bus clock) is controlled through this register. The I/O impedance of PCI signals is controlled by POR configuration pins (described in Section 4.4.3.12,

"PCI I/O Impedance"). *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* provides exact I/O impedances.

Table 18-6 describes PORIMPSCR fields.

Table 18-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–13	—	Reserved
14	PCI2_Z	PCI2 I/O impedance 0 Low impedance 1 High impedance
15	PCI1_Z	PCI1 I/O impedance (lower 32 bits when in 64-bit mode) as set by POR signals (see Section 4.4.3.12, "PCI I/O Impedance"). 0 Low impedance 1 High impedance
16–24	—	Reserved
25	LALE_Z	I/O impedance for local bus address latch enable 0 Low impedance 1 High impedance
26	LADP_Z	I/O impedance for local bus address/data and data parity (LAD[0:31] and LDP[0:7]) 0 Low impedance 1 High impedance
27	LA_CKE_Z	I/O impedance for local bus address and clock enable (LA[27:31] and LCKE) 0 Low impedance 1 High impedance
28	LCS12_Z	I/O impedance for two local bus chip selects ($\overline{\text{LCS}}$1 and $\overline{\text{LCS}}$2 only) Other chip selects use a fixed high I/O impedance 0 Low impedance 1 High impedance
29	LWE_Z	I/O impedance for local bus write enables ($\overline{\text{LWE}}$[0:3]) 0 Low impedance 1 High impedance
30	LGPL_Z	I/O impedance for local bus general-purpose lines (LGPL[0:5]) 0 Low impedance 1 High impedance
31	LCLK_Z	I/O impedance for local bus clocks (LCLK[0:2]) 0 Low impedance 1 High impedance

18.4.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 18-4, PORDEVSR reports other POR settings for I/O devices as described in Section 4.4.3.7, "TSEC Width," Section 4.4.3.8, "TSEC1 Protocol," Section 4.4.3.9, "TSEC2 Protocol," Section 4.4.3.13, "PCI Arbiter Configuration," and Section 4.4.3.11, "PCI Width Configuration."

	0	1				5	6	7	8				12	13		14	15	16															31
R	ECW	0	0	0	0	0	ECP		0	0	0	0	0	PCI2_ARB		PCI1_ARB	PCI32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																	
Reset	*n*000_00*nn*_0000_0*nnn*_0000_0000_0000_0000																																
Offset	0xE_000C																																

Figure 18-4. POR Device Status Register (PORDEVSR)

Table 18-7 describes the bit settings of PORDEVSR.

Table 18-7. PORDEVSR Field Descriptions

Bits	Name	Description
0	ECW	Gigabit Ethernet controller width 0 Reduced (RGMII, RTBI) 1 Full (MII, GMII, TBI)
1–5	—	Reserved
6–7	ECP	Gigabit Ethernet controller protocol 00 Both TSEC blocks use a media Independent Interface protocols (MII/GMII/RGMII) 01 TSEC1 uses MII, TSEC2 uses TBI 10 TSEC1 uses TBI, TSEC2 uses MII 11 Both TSEC blocks use a ten bit interface protocol (TBI or RTBI)
8–12	—	Reserved
13	PCI2_ARB	PCI2 arbiter enable 0 PCI2 arbiter is disabled 1 PCI2 arbiter is enabled
14	PCI1_ARB	PCI1 arbiter enable 0 PCI1 arbiter is disabled 1 PCI1 arbiter is enabled
15	PCI32	PCI1 interface width 0 64-bit 1 32-bit. PCI2 may be enabled.
16–31	—	Reserved

18.4.1.5 POR Debug Mode Status Register (PORDBGMSR)

PORDBGMSR, shown in Figure 18-5, holds debug mode settings from the POR configuration pins as described in Section 4.4.3.15, “Memory Debug Configuration,” Section 4.4.3.16, “DDR Debug Configuration,” and Section 4.4.3.14, “PCI Debug Configuration.”

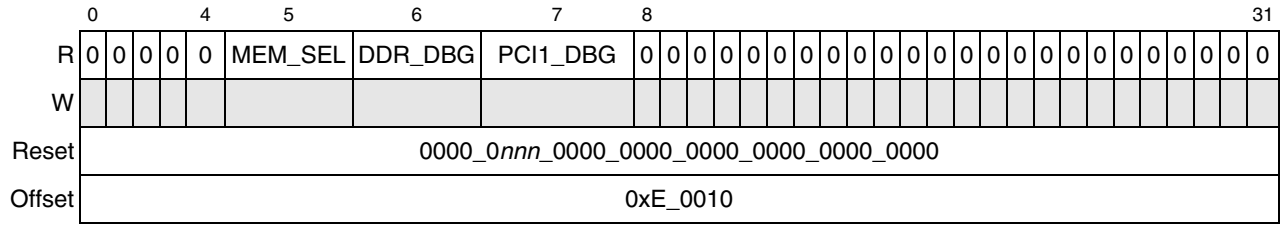


Figure 18-5. POR Debug Mode Status Register (PORDBGMSR)

Table 18-8 describes the bit settings of PORDBGMSR.

Table 18-8. PORDBGMSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	MEM_SEL	Memory select. Indicates which controller is driving MSRCID[0:4] and MDVAL 0 Local bus controller is driving debug information 1 DDR SDRAM controller is driving debug information
6	DDR_DBG	DDR debug configuration 0 Source ID and data valid information is being driven on ECC pins of DDR SDRAM interface 1 Normal mode. ECC information is being driven on ECC pins of DDR SDRAM interface
7	PCI1_DBG	PCI1 debug configuration 0 PCI1 drives source ID onto address signals PCI1_AD[62:58]. Note that PCI1 must be configured for 64-bit mode to see source ID debug information. 1 PCI1 drives address onto address signals PCI1_AD[62:58]. (See Section 20.4.2, “PCI Interface Debug,” for additional details.)
8–31	—	Reserved

18.4.1.6 General-Purpose POR Configuration Register (GPPORCR)

Shown in Figure 18-6, GPPORCR stores the value sampled from the local bus address/data signals, LAD[0:31], during POR, as described in Section 4.4.3.19, “General-Purpose POR Configuration.” Software can use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

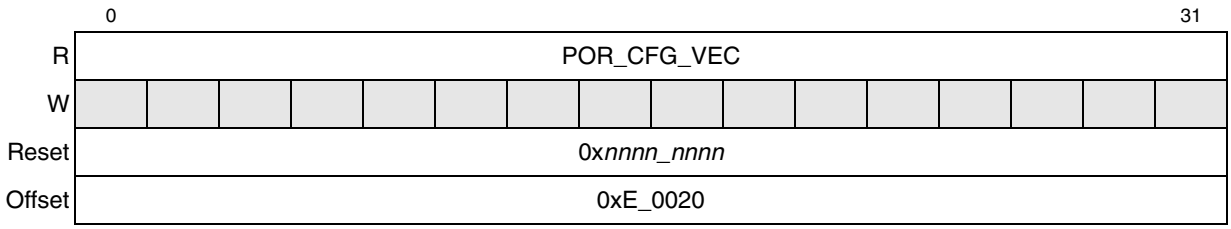


Figure 18-6. POR Configuration Register (GPPORCR)

Table 18-9 describes the bit settings of GPPORCR.

Table 18-9. GPPORCR Field Descriptions

Bits	Name	Description
0–31	POR_CFG_VEC	General-purpose POR configuration vector sampled from local bus address/data signals at the negation of HRESET. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.

18.4.1.7 General-Purpose I/O Control Register (GPIOCR)

Shown in Figure 18-7, GPIOCR contains the enable bits for each group of pins that may be used for general-purpose I/O. These bits have meaning only if the pins are not being used for their primary function. Note that when these signals are enabled as general-purpose I/O signals, they are read and written through GPINDR and GPOUTDR described in Section 18.4.1.9, "General-Purpose Input Data Register (GPINDR)," and Section 18.4.1.8, "General-Purpose Output Data Register (GPOUTDR)." Section 18.5.2, "General-Purpose I/O Signals," describes the use of general-purpose I/O signals.

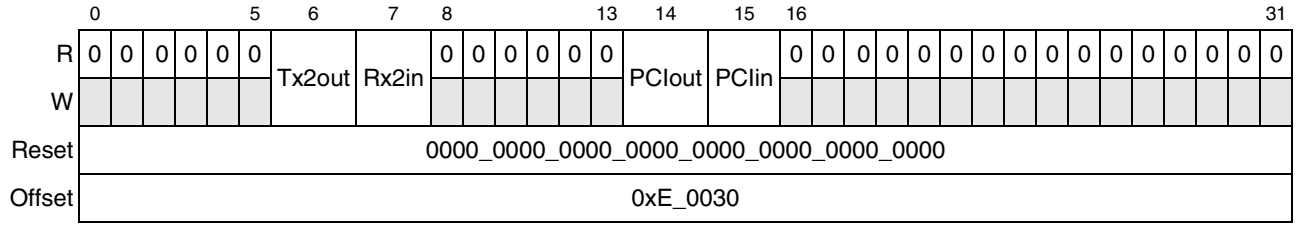


Figure 18-7. General-Purpose I/O Control Register (GPIOCR)

Table 18-10 describes the bit settings of GPIOCR.

Table 18-10. GPIOCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	Tx2out	Enables TSEC2_Tx[0:7] for use as general-purpose outputs if the TSEC2 interface is disabled. (See Section 18.4.1.11, "Device Disable Register (DEVDISR).") 0 TSEC2_Tx[0:7] function as described in the TSEC chapter. 1 TSEC2_Tx[0:7] function as general-purpose outputs.
7	Rx2in	Enables TSEC2_Rx[0:7] for use as general-purpose inputs if the TSEC2 interface is disabled. (See Section 18.4.1.11, "Device Disable Register (DEVDISR).") 0 TSEC2_Rx[0:7] function as described in the TSEC chapter. 1 TSEC2_Rx[0:7] function as general-purpose inputs.
8–13	—	Reserved
14	PCIout	Enables PCI2_AD[15:8] for use as general-purpose outputs. Note that the PCI1 interface must be configured for 32-bit mode AND the PCI2 interface must be disabled. (See Section 18.4.1.11, "Device Disable Register (DEVDISR).") 0 PCI2_AD[15:8] function as described in the PCI chapter. 1 PCI2_AD[15:8] function as general-purpose outputs.

Table 18-10. GPIOCR Field Descriptions (continued)

Bits	Name	Description
15	PCIin	Enables PCI2_AD[7:0] for use as general-purpose inputs. Note that the PCI1 interface must be configured for 32-bit mode AND the PCI2 interface must be disabled. (See Section 18.4.1.11, "Device Disable Register (DEVDISR).") 0 PCI2_AD[7:0] function as described in the PCI chapter. 1 PCI2_AD[7:0] function as general-purpose inputs.
16–31	—	Reserved

18.4.1.8 General-Purpose Output Data Register (GPOUTDR)

GPOUTDR, shown in Figure 18-8, contains the data driven as general-purpose output on TSEC2_TXD[0:7] and/or PCI2_AD[15:8] when either of these buses is configured as a general-purpose I/O bus, as described in Section 18.4.1.7, "General-Purpose I/O Control Register (GPIOCR)." Writes to GPOUTDR affect only pins enabled as general-purpose outputs. Reads return valid data only for bits corresponding to pins enabled as general-purpose outputs. GPOUTDR may be accessed using single byte writes (using big-endian addressing) so that writes to one byte do not affect outputs controlled by others.

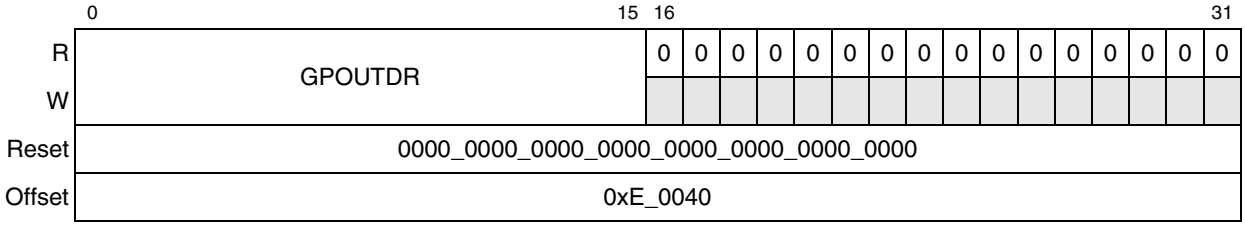


Figure 18-8. General-Purpose Output Data Register (GPOUTDR)

Table 18-11 describes the fields of GPOUTDR.

Table 18-11. GPOUTDR Field Descriptions

Bits	Name	Description
0–15	GPOUTDR	General-purpose output data. When the corresponding signals are configured to be general-purpose output signals, the values of the bits of GPOUTDR are driven onto those pins. GPOUTDR[0:7] corresponds to TSEC2_TXD[0:7] and GPOUTDR[8:15] corresponds to PCI2_AD[15:8] as follows: GPOUTDR[0] ↔ TSEC2_TxD[0] GPOUTDR[1] ↔ TSEC2_TxD[1] GPOUTDR[2] ↔ TSEC2_TxD[2] GPOUTDR[3] ↔ TSEC2_TxD[3] GPOUTDR[4] ↔ TSEC2_TxD[4] GPOUTDR[5] ↔ TSEC2_TxD[5] GPOUTDR[6] ↔ TSEC2_TxD[6] GPOUTDR[7] ↔ TSEC2_TxD[7] GPOUTDR[8] ↔ PCI2_AD[15] GPOUTDR[9] ↔ PCI2_AD[14] GPOUTDR[10] ↔ PCI2_AD[13] GPOUTDR[11] ↔ PCI2_AD[12] GPOUTDR[12] ↔ PCI2_AD[11] GPOUTDR[13] ↔ PCI2_AD[10] GPOUTDR[14] ↔ PCI2_AD[9] GPOUTDR[15] ↔ PCI2_AD[8]
16–31	—	Reserved, should be cleared

18.4.1.9 General-Purpose Input Data Register (GPINDR)

GPINDR, shown in Figure 18-9, contains the data currently sampled as general-purpose input on TSEC2_RXD[0:7] and/or PCI2_AD[7:0] when either of these buses is configured as a general-purpose I/O bus in GPIOCR. (See Section 18.4.1.7, “General-Purpose I/O Control Register (GPIOCR).”) GPINDR bits are updated only if the associated bits are configured as general-purpose input pins rather than their primary functions.

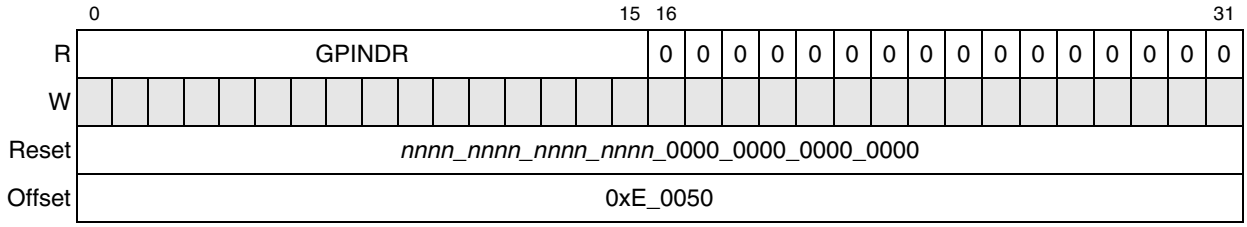


Figure 18-9. General-Purpose Input Data Register (GPINDR)

Table 18-12 describes the fields of GPINDR.

Table 18-12. GPINDR Field Descriptions

Bits	Name	Description
0–15	GPINDR	General-purpose input data. When the corresponding signals are configured to be general-purpose input signals, the values sampled on these signals are reflected in GPINDR. GPINDR[0:7] corresponds to TSEC2_RxD[0:7] and GPINDR[8:15] corresponds to PCI2_AD[7:0] as follows: GPINDR[0] ↔ TSEC2_RxD[0] GPINDR[1] ↔ TSEC2_RxD[1] GPINDR[2] ↔ TSEC2_RxD[2] GPINDR[3] ↔ TSEC2_RxD[3] GPINDR[4] ↔ TSEC2_RxD[4] GPINDR[5] ↔ TSEC2_RxD[5] GPINDR[6] ↔ TSEC2_RxD[6] GPINDR[7] ↔ TSEC2_RxD[7] GPINDR[8] ↔ PCI2_AD[7] GPINDR[9] ↔ PCI2_AD[6] GPINDR[10] ↔ PCI2_AD[5] GPINDR[11] ↔ PCI2_AD[4] GPINDR[12] ↔ PCI2_AD[3] GPINDR[13] ↔ PCI2_AD[2] GPINDR[14] ↔ PCI2_AD[1] GPINDR[15] ↔ PCI2_AD[0]
16–31	—	Reserved

18.4.1.10 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 18-10, PMUXCR contains bits that enable DMA channels 2 and 3, which exist as alternate functions on local bus chip select pins $\overline{\text{LCS}}$[5:7], and interrupt input pins IRQ[9:11], respectively. Specifically, DMA request, acknowledge, and done signals comprise the secondary functions for the associated IRQ and local bus chip select signals. Note that CPM signal multiplexing is handled separately through the CPM programming model described in Chapter 45, “Parallel I/O Ports.”

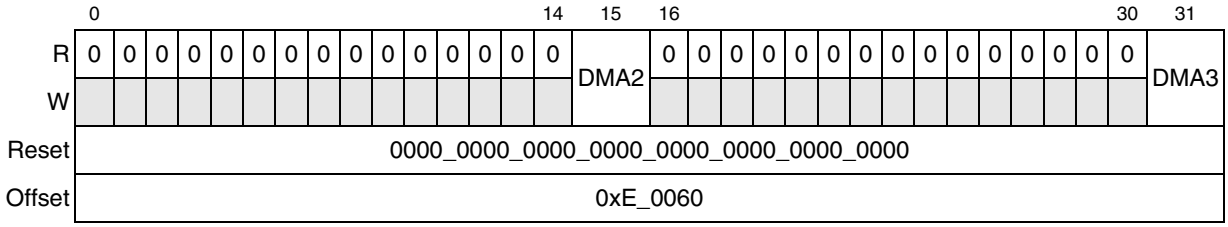


Figure 18-10. Alternate Function Pin Multiplex Control Register (PMUXCR)

Table 18-13 describes the bit settings of PMUXCR.

Table 18-13. PMUXCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	DMA2	Enables DMA channel 2 signals 0 DMA channel 2 is not exposed to pins; the pins retain their primary function as local bus chip selects. 1 DMA channel 2 is exposed to pins as follows: LCS5 functions as DMA_DREQ2 LCS6 functions as DMA_DACK2 LCS7 functions as DMA_DDONE2
16–30	—	Reserved
31	DMA3	Enables DMA channel 3 signals 0 DMA channel 3 is not exposed to pins; the pins retain their primary function as interrupt requests. 1 DMA channel 3 is exposed to pins as follows: IRQ9 functions as DMA_DREQ3 IRQ10 functions as DMA_DACK3 IRQ11 functions as DMA_DDONE3

18.4.1.11 Device Disable Register (DEVDISR)

DEVDISR, shown in Figure 18-11, contains disable bits for various MPC8555E functional blocks.

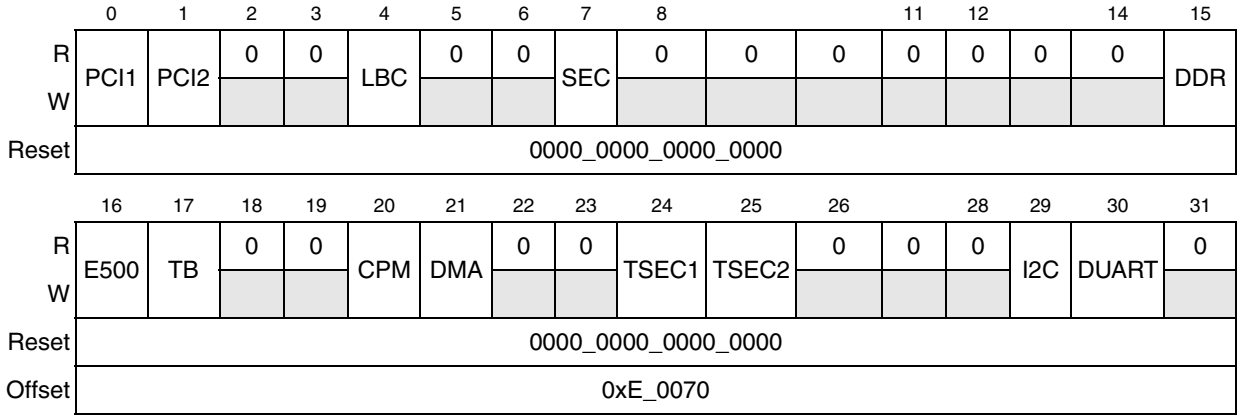


Figure 18-11. Device Disable Register (DEVDISR)

All functional blocks are enabled after reset; unneeded blocks can be disabled to reduce power consumption or allow their signals to be used as general-purpose I/O signals. See Section 18.4.1.7, "General-Purpose I/O Control Register (GPIOCR)." Blocks disabled by DEVDISR must not be re-enabled without a hard reset. Section 18.5.1.4, "Shutting Down Unused Blocks," has more information on the use of DEVDISR. Table 18-14 describes DEVDISR fields.

Table 18-14. DEVDISR Field Descriptions

Bits	Name	Description
0	PCI1	PCI1 controller disable 0 PCI1 controller enabled 1 PCI1 controller disabled
1	PCI2	PCI2 controller disable 0 PCI2 controller enabled 1 PCI2 controller disabled. PCI2_AD[15:0] may be used as general-purpose I/O provided PCI1 is in 32-bit mode.
2–3	—	Reserved
4	LBC	Local bus controller disable 0 Local bus controller enabled 1 Local bus controller disabled
5–6	—	Reserved
7	SEC	Security disable 0 Security enabled 1 Security disabled
8–11	—	Reserved
12–14	—	Reserved
15	DDR	DDR SDRAM controller disable 0 DDR SDRAM controller enabled 1 DDR SDRAM controller disabled
16	E500	e500 core disable 0 e500 core enabled 1 e500 core disabled. Places the core in the core_stopped state in which it does not respond to interrupts. Equivalent to nap mode. Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core including the timer facilities. For more information, see Section 18.5.1.4, “Shutting Down Unused Blocks.”
17	TB	Time base (timer facilities) of the e500 core disable 0 Timer facilities enabled 1 Timer facilities disabled
18–19	—	Reserved
20	CPM	Communications processor module disable 0 CPM enabled 1 CPM disabled
21	DMA	DMA controller disable 0 DMA controller enabled 1 DMA controller disabled
22–23	—	Reserved
24	TSEC1	Three-speed Ethernet controller 1 disable 0 TSEC1 enabled 1 TSEC1 disabled
25	TSEC2	Three-speed Ethernet controller 2 disable 0 TSEC2 enabled 1 TSEC2 disabled. RxD and TxD pins may be used for general-purpose I/O.

Table 18-14. DEVDISR Field Descriptions (continued)

Bits	Name	Description
26–28	—	Reserved
29	I2C	I^2C controller disable 0 I^2C controller enabled 1 I^2C controller disabled
30	DUART	Dual UART controller disable 0 DUART enabled 1 DUART disabled
31	—	Reserved

18.4.1.12 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 18-12, POWMGTCSR contains bits for placing the MPC8555E into low power states and for controlling when it wakes up. It also contains power management status bits. See Section 18.5.1.8.2, "Interrupts and Power Management Controlled by POWMGTCSR," for more information.

	0	1	2	11	12	13	14	15	27	28	29	30	31
R	IRQ_MSK	CI_MSK	0 0 0 0 0 0 0 0 0 0	DOZ	0	SLP	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			DOZING	NAPPING	SLPING	0
W													
Reset	0000_0000_0000_0000_0000_0000_0000_0000												
Offset	0xE_0080												

Figure 18-12. Power Management Control and Status Register (POWMGTCSR)

Table 18-15 describes the bit settings of POWMGTCSR.

Table 18-15. POWMGTCSR Field Descriptions

Bits	Name	Description
0	IRQ_MSK	Interrupt input mask 0 Interrupts cause the device to wake up from a low-power state. 1 Interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of an interrupt request.
1	CI_MSK	Critical interrupt input mask 0 Critical interrupts cause the device to wake up from a low power state. 1 Critical interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of a critical interrupt.
2–11	—	Reserved
12	DOZ	Doze mode 0 No request to put device in doze mode. Note that this bit is automatically cleared on MCP, UDE, $\overline{\text{SRESET}}$, *core_tbint* (from the core) and also *int* and *cint* if not masked. 1 Device is to be placed in doze mode. Instruction fetching is halted in the e500 core. Note that this bit is logically ORed with HID0[DOZE].
13	—	Reserved

Table 18-15. POWMGTCSR Field Descriptions (continued)

Bits	Name	Description
14	SLP	Sleep mode 0 No request to put device in sleep mode 1 Device is to be placed in sleep mode. Instruction fetching is halted, snooping of L1 caches is disabled, and most functional blocks are shut down in both the e500 core and the system logic.
15–27	—	Reserved
28	DOZING	Doze status 0 Device is not in doze mode 1 The MPC8555E is in doze mode because POWMGTCSR[DOZ] is set or because HID0[DOZE] and MSR[WE] (in the e500 core) are set. The core has halted instruction fetching, but all other functional blocks in the core and device are fully operational.
29	NAPPING	Nap status 0 Device is not in nap mode. 1 The MPC8555E is in nap mode because HID0[NAP] and MSR[WE] are set. The core has halted instruction fetching, snooping of the L1 caches is disabled, and all of the core's functional units except the timer facilities are shut down. All functional blocks in the device are fully operational.
30	SLPING	Sleep status 0 Device is not attempting to reach sleep mode 1 The device is attempting to SLEEP because POWMGTCSR[SLP] is set or because HID0[SLEEP] and MSR[WE] (in the e500 core) are set. Most functional blocks in the core and device are shut down or are attempting to shut down.
31	—	Reserved. Should be cleared.

18.4.1.13 Machine Check Summary Register (MCPSUMR)

Shown in Figure 18-13, MCPSUMR contains bits summarizing some of the sources of a pending machine check interrupt. All MCPSUMR bits function as write-1-to-clear.

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

Note that other conditions can cause a machine check condition not summarized in MCPSUMR. For example, uncorrectable read errors cause the assertion of *core_fault_in*, which may directly cause a machine check (if HID1[RFXE] = 1). If RFXE = 0, the assertion of *core_fault_in* does not directly cause a machine check interrupt, but must be handled by the block that generated the error. For more information about RFXE, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."

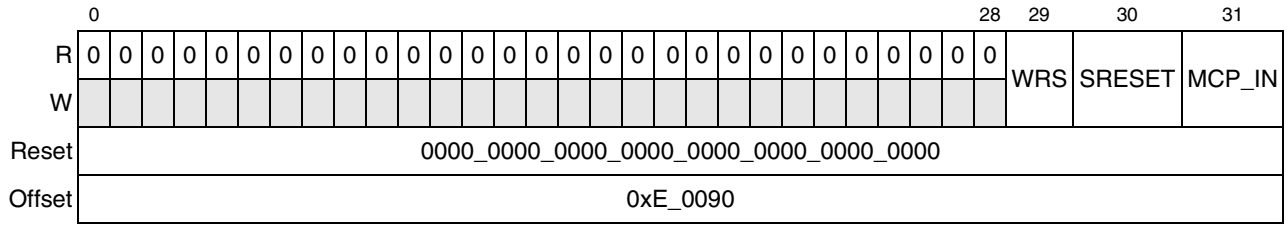


Figure 18-13. Machine Check Summary Register (MCPSUMR)

Table 18-16 describes the bit settings of MCPSUMR.

Table 18-16. MCPSUMR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	WRS	Watchdog timer machine check 0 Machine check exception was not caused by watchdog timer 1 Machine check was caused by a soft reset condition from the e500 watchdog timer as configured in the core's TSR. Specifically, TSR[WRS] = 01 and a watchdog reset condition occurred.
30	SRESET	Soft reset machine check 0 Machine check exception was not caused by $\overline{\text{SRESET}}$ assertion 1 Machine check exception was caused by the assertion of the $\overline{\text{SRESET}}$ input signal
31	MCP_IN	$\overline{\text{MCP}}$ signal asserted 0 Machine check exception was not caused by $\overline{\text{MCP}}$ assertion 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP}}$ input signal

18.4.1.14 Processor Version Register (PVR)

Shown in Figure 18-14, the PVR contains the e500 processor version number. It is a memory-mapped copy of the PVR in the e500 core (and is therefore accessible to external devices). See Section 6.5.3, "Processor Version Register (PVR)." Section 5.2, "e500 Processor and System Version Numbers," lists the complete values for the MPC8555E

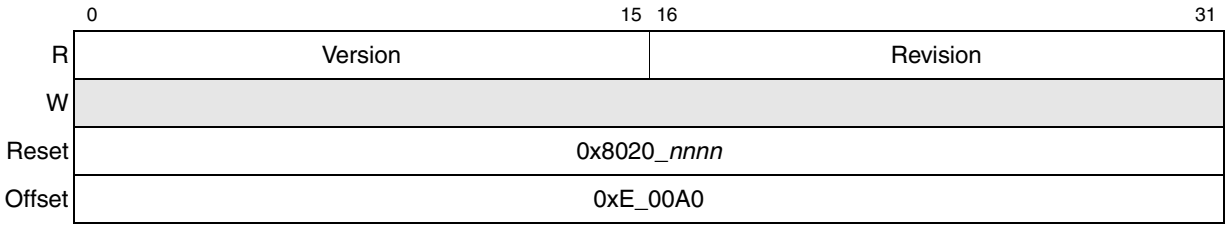


Figure 18-14. Processor Version Register (PVR)

Table 18-17 describes the fields of PVR.

Table 18-17. PVR Field Descriptions

Bits	Name	Description
0–15	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported.
16–31	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level.

18.4.1.15 System Version Register (SVR)

Shown in Figure 18-15, the SVR contains the system version number for the MPC8555E/MPC8541E implementation. This value can also be read though the SVR SPR of the e500 core. See Section 6.5.4, "System Version Register (SVR)." Section 5.2, "e500 Processor and System Version Numbers," lists the complete values for the MPC8555E.

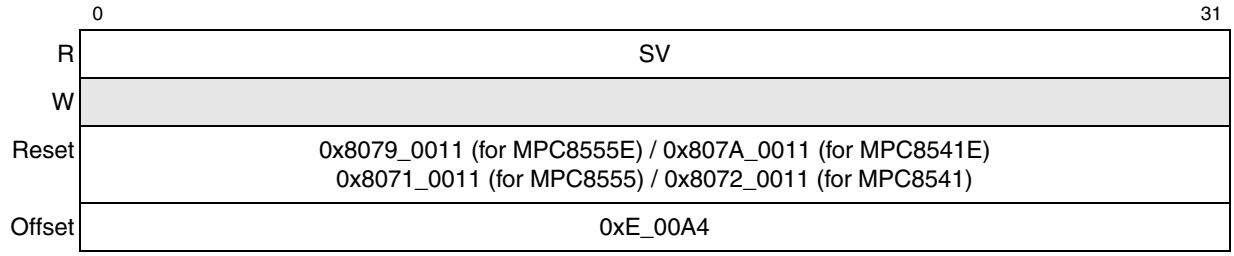


Figure 18-15. System Version Register (SVR)

Table 18-18 describes the fields of SVR.

Table 18-18. SVR Field Descriptions

Bits	Name	Description
0–31	SV	System version: 0x8079_0011 (for MPC8555E) / 0x807A_0011 (for MPC8541E) 0x8071_0011 (for MPC8555) / 0x8072_0011 (for MPC8541)

18.4.1.16 Clock Out Control Register (CLKOCR)

Shown in Figure 18-16, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK_OUT) signal.

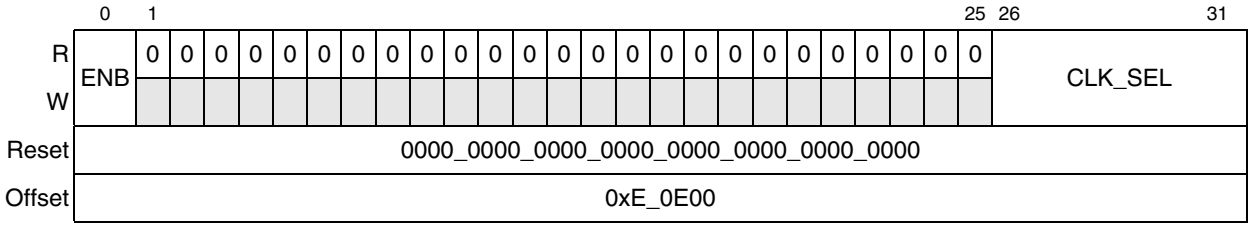


Figure 18-16. Clock Out Control Register (CLKOCR)

Table 18-19 describes the bit settings of CLKOCR.

Table 18-19. CLKOCR Field Descriptions

Bits	Name	Description
0	ENB	Clock out enable 0 CLK_OUT signal is three-stated 1 CLK_OUT signal is driven according to CLKOCR[CLK_SEL]

Table 18-19. CLKOCR Field Descriptions (continued)

Bits	Name	Description
1–25	—	Reserved
26–31	CLK_SEL	Clock out select 000000 CCB (platform) clock 000001 CCB (platform) clock divided by 2 000010 SYSCLK (echoes SYSCLK input) 000011 SYSCLK divided by 2 (demonstrates platform PLL lock) 000100 Reserved 000101 Reserved 000110 Reserved 000111 Reserved 001000 Reserved 001001 Reserved 001010 Reserved 001011 Reserved 001100 Reserved 001101 Reserved 001110 Reserved 001111 Reserved 01xx0x Reserved 01xx1x Reserved 10x000 Reserved 10x001 Reserved 10x010 PCI1 bus clock 10x011 PCI1 bus clock divided by 2 10x100 Reserved 10x101 Reserved 10x110 Reserved 10x111 Logic 0 11x000 Reserved 11x001 Reserved 11x010 PCI2 bus clock 11x011 PCI2 bus clock divided by 2 11x100 Reserved 11x101 Reserved 11x110 Reserved 11x111 Logic 1

18.4.1.17 Local Bus DLL Control Register (LBDLLCR)

Shown in Figure 18-17, the LBDLLCR contains control bits that allow debug of the local bus controller's DLL. The delay chain of the DLL is made up of 128 tap points.

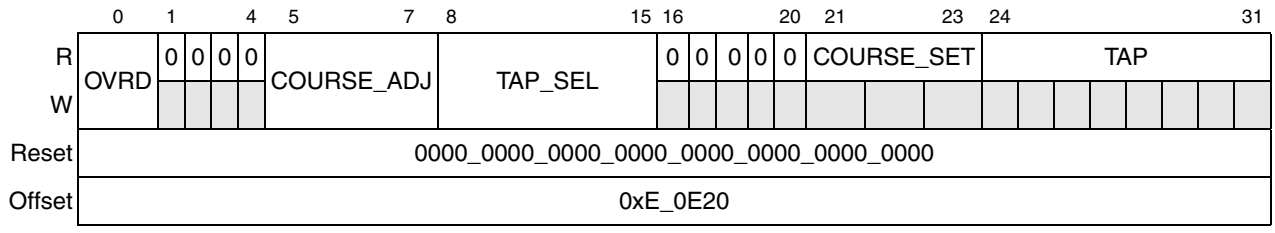


Figure 18-17. Local Bus DLL Control Register (LBDLLCR)

Table 18-20 describes the bit settings of LBDLLCR.

Table 18-20. LBDLLCR Field Descriptions

Bits	Name	Description
0	OVRD	Override mode 0 Override mode disabled 1 Override of current delay chain tap point with the TAPSEL tap point enabled
1–4	—	Reserved
5–7	COURSE_ADJ	Course adjustment value to be used by the DLL when in override mode (OVRD = 1). The course adjustment is the number of CCB clock cycles of delay to inject before the delay chain. When leaving override mode (OVRD cleared) this course adjust point serves as the starting point for a dynamic search for a lock point.
8–15	TAP_SEL	TAP select value to be used by the DLL when in override mode (OVRD = 1). Selects the tap point within the delay chain. When leaving override mode (OVRD cleared) this tap point serves as the starting point for a dynamic search for a lock point.
16–20	—	Reserved
21–23	COURSE_SET	Reports the current course delay setting found by the dynamic search algorithm that produced a lock. Measured in CCB clock cycles
24–31	TAP	Reports the tap value found by the dynamic search algorithm that produced a lock. Measured in tap points

18.5 Functional Description

This section describes the global utilities from a functional perspective.

18.5.1 Power Management

The MPC8555E 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 (DEVDISR). Additionally, software executing on the e500 core can access the core's SPRs to put the device into doze, nap, or sleep power down state. Finally, software can access a memory-mapped register (POWMGTCR) in the global utilities block to put the device in the doze or sleep states.

Note that the software that writes to either DEVDISR or POWMGTCR can be executing either on the e500 core or on an external master that can write to the MPC8555E memory-mapped registers through the PCI interfaces.

These features are described in further detail in this section.

18.5.1.1 Relationship Between Core and Device Power Management States

The MPC8555E has three low-power states: doze, nap, and sleep. The mapping of core and device power management states is shown in Figure 18-18 showing state transitions from the perspective of the e500 core.

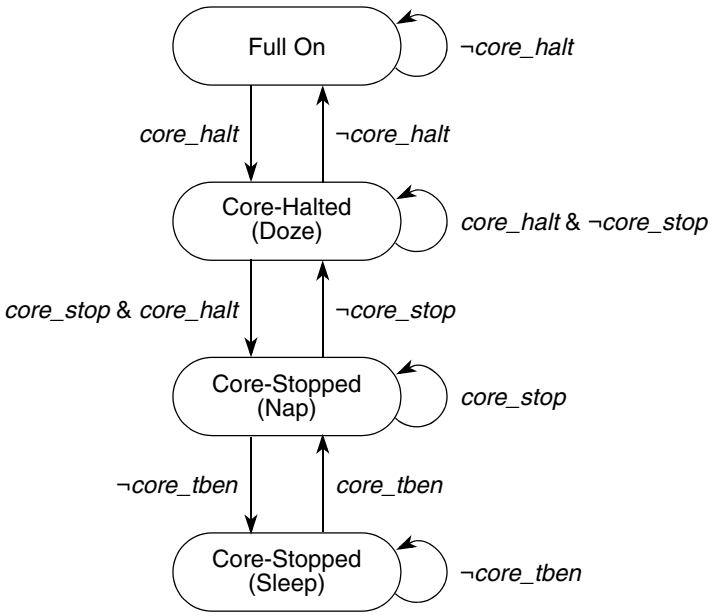


Figure 18-18. e500 Core Power Management State Diagram

For each operating state represented in the diagram, the core's state is listed first, with the corresponding state of the MPC8555E shown beneath it in parenthesis. Note that there are many other variables that control the state transitions between MPC8555E power management states. These additional variables are described in more detail in Section 18.5.1.7, "Power-Down Sequence Coordination."

Table 18-21 lists basic characteristics of the low-power modes and the full on mode.

Table 18-21. MPC8555E Power Management Modes—Basic Descriptions

Mode	Description	Core Responds To		Signal States	
		Snoop	Interrupts	READY	ASLEEP
Full On	All units operating normally	Yes	Yes	Asserted	Negated
Doze	Core stops dispatching new instructions (core is halted)	Yes	Yes	Negated	Negated
Nap	Core is stopped with clocks off except to time base Should flush data cache before entering	No	Yes	Negated	Negated
Sleep	Core is stopped with clocks off. Clocks powered down to all blocks (including core time base) except to the interrupt controller (PIC) unit	No	Yes	Negated	Asserted

18.5.1.2 $\overline{\text{CKSTP\_IN}}$ Is Not Power Management

$\overline{\text{CKSTP\_IN}}$ is not described here because it is not considered a power management signal, although asserting it does stop the core and a stopped core is technically in a low-power mode. $\overline{\text{CKSTP\_IN}}$ is described in Section 18.3.2, "Detailed Signal Descriptions."

18.5.1.3 Dynamic Power Management

Many blocks in the MPC8555E can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

18.5.1.4 Shutting Down Unused Blocks

As described in Section 18.4.1.11, "Device Disable Register (DEVDISR)," DEVDISR provides a way to shut down certain functional blocks within the MPC8555E when they are not needed in a particular system. DEVDISR can be written by the e500 core or by an external master. Powering down a block in this way turns off all clocks to that block.

DEVDISR was designed with the expectation that, once initialized by software, it would be modified only by a hard system reset ($\overline{\text{HRESET}}$). It is recommended that this register be written only during system initialization. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. (Setting DEVDISR[TB] disables the core's timer facilities, and setting DEVDISR[E500] places the core in the core_stopped state in which it does not respond to interrupts.) The results of re-enabling previously disabled blocks (by clearing the corresponding DEVDISR field) without a hard reset are boundedly undefined.

NOTE

Functional blocks disabled using DEVDISR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

18.5.1.5 Software-Controlled Power-Down States

The e500 software can place the device in doze, nap, or sleep power-down states by writing to HID0 in the core. In addition, external masters can write to the memory-mapped POWMGTCR in the MPC8555E to cause the device to enter doze or sleep modes.

18.5.1.5.1 Doze Mode

In doze mode, the e500 core suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1 data cache is still supported and thus the data in the data cache is kept coherent. Interrupts directed to the core as described in Section 10.1.3, "Interrupts to the Processor Core," are monitored by the device and cause the MPC8555E to use the defined handshake mechanism to exit the core from doze mode to allow the core to recognize and process the interrupt; however, unless the interrupt subroutine turns off (or masks) the control bits that enabled doze mode (MSR[WE], and HID0[DOZE]), the device re-enters doze mode after the interrupt has been serviced. See Section 18.5.1.8, "Interrupts and Power Management," for more information.

The e500 core's timer facilities are still enabled during doze mode, and core time base interrupts can be generated. All device logic external to the core remains fully operational in doze mode.

18.5.1.5.2 Nap Mode

In nap mode all clocks internal to the e500 core are turned off except for its timer facilities clock (the core time base). The L1 caches do not respond to snoops in nap mode, so if coherency with external I/O transactions is required, the L1 cache must be flushed before entering nap mode.

Similar to doze mode, interrupts occurring in nap mode cause the device to wake up the e500 core in order to service the interrupt. However, unless the interrupt service routine changes the control bits that caused the device to enter nap mode (MSR[WE], and HID0[NAP]), the MPC8555E returns to nap mode after the interrupt is serviced. See Section 18.5.1.8, "Interrupts and Power Management," for more information.

All device logic external to the e500 core remains fully operational in nap mode.

18.5.1.5.3 Sleep Mode

In sleep mode, all clocks internal to the e500 core are turned off, including the timer facilities clock. All I/O interfaces in the device logic are also shut down. Only the clocks to the MPC8555E PIC are still operational so that an external interrupt can wake up the device. Note that the DDR controller does not shut down unless DDR_SDRAM_INTERVAL[REFINT] is set to a non-zero value. See Section 9.4.1.7, "DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)," for details.Note that external interrupts from port C of the CPM are a special case and do not reach the PIC when the device is asleep. Therefore, they do not cause the device to wake up.

After the core and I/O interfaces have shut down, ASLEEP is asserted and READY is negated.

NOTE

Only external interrupts can wake the MPC8555E from sleep mode. Internal interrupt sources like the core interval timer or watchdog timer depend on an active clock for their operation and these are disabled in sleep mode.

18.5.1.6 Power Management Control Fields

The e500 core provides the following fields to signal power management requests to the MPC8555E device logic.

- MSR[WE]—Used to qualify the values of HID0[DOZE,NAP,SLEEP] in the generation of the internal *doze*, *nap*, and *sleep* signals
- HID0[DOZE]—Signals the MPC8555E to initiate doze mode
- HID0[NAP]—Signals the MPC8555E to initiate nap mode
- HID0[SLEEP]—Signals the MPC8555E to initiate sleep mode

These register fields and their functional relationship are shown in Section 6.10.1, "Hardware Implementation-Dependent Register 0 (HID0)," and Section 6.5.1, "Machine State Register (MSR)." The *e500 Core Family Reference Manual* has details on accessing these power management control bits.

An external master can also initiate power management requests by setting the DOZ or SLP bits in the memory-mapped power management control and status register (POWMGTCSR). Because the core responds to snoops while dozing but not while napping, maintaining cache coherency requires significant

preparation by the core before entering nap mode. For this reason only the core can initiate a nap during normal operation while other masters can initiate a doze.

18.5.1.7 Power-Down Sequence Coordination

To preserve cache coherency and otherwise avoid loss of system state, the core's transition to low-power modes is coordinated by a set of handshaking signals, shown in Figure 18-19, and protocols with all other MPC8555E functional blocks that respond to power-down requests. The mode-transition protocol is executes automatically under these conditions and is shown in Figure 18-18 and described in Table 18-22.

The column in Table 18-22 showing the global utilities block as initiating a low-power mode corresponds to the external masters that can write to the POWMGTCSR that resides in the global utilities block. For the MPC8555E, these are the PCI interfaces. However, note that the core can also write to POWMGTCSR and, in this case, can initiate power management through the global utilities block.

Table 18-22. Power Management Entry Protocol and Initiating Functional Units

Low-Power Mode	Entry Protocol	Initiating Functional Unit	
		Global Utilities	Core
Doze	1. Assert *core_halt* input to core. 2. Wait for *core_halted* handshake from core.	√	√
Nap	1. Follow doze protocol 2. Assert *core_stop* input to core. 3. Wait for *core_stopped* handshake from core.	—	√
Sleep	1. Follow doze protocol; send stop requests to rest of device. 2. Follow nap protocol. 3. Wait for all interfaces to acknowledge stop requests. 4. Assert ASLEEP, negate READY, power down all clocks except to PIC unit.	√	√

As shown in Figure 18-19, the e500 core enters low-power modes only in response to the *core_halt*, *core_stop*, or *core_tben* inputs from the MPC8555E power management logic. These inputs may be prompted by the core (by setting the NAP, DOZE, or SLEEP bits in the HID0 when enabled by setting MSR[WE]) or by an external master (by setting POWMGTCSR[DOZ,SLP]).

Figure 18-19 shows how all the clocking to the core timer facilities is disabled by clearing HID0[TBEN]. When enabled, (HID0[TBEN] = 1), the clock source is either the CCB clock divided by eight (the default) or a synchronized version of the RTC input. For more details, see Section 6.10.1, "Hardware Implementation-Dependent Register 0 (HID0)."

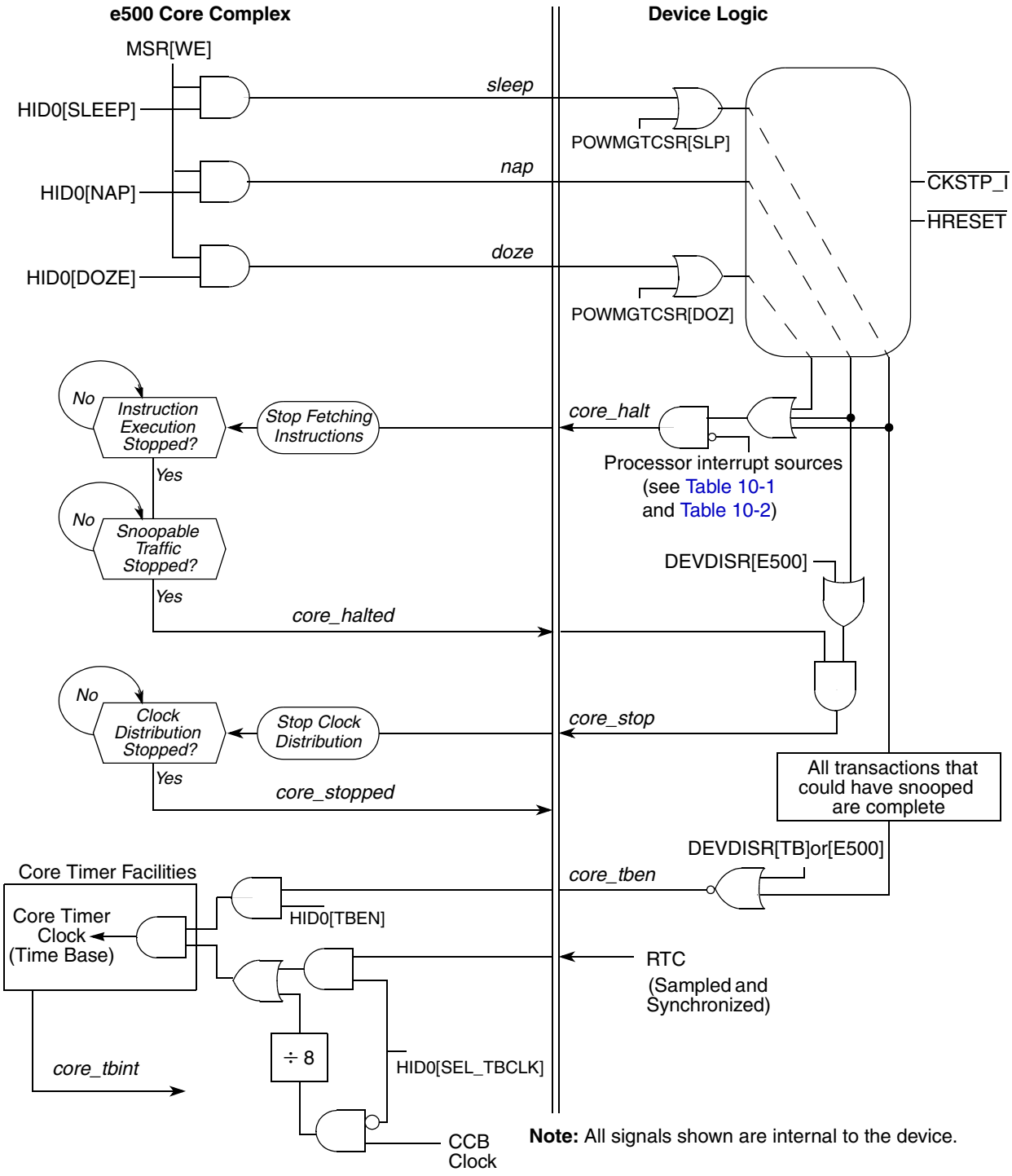


Figure 18-19. MPC8555E Power Management Handshaking Signals

18.5.1.8 Interrupts and Power Management

Whether low-power modes are automatically re-enabled after an interrupt is processed differs depending on whether the low power mode was entered due to a write to the core MSR[WE] bit or the low power mode was entered due to a write to POWMGTCSR.

18.5.1.8.1 Interrupts and Power Management Controlled by MSR[WE]

When an interrupt is asserted to the CPU, the core complex saves portions of the MSR to MCSRR1, CSRR1, or SRR1 (depending on the type of interrupt), and restores those values on return from the routine. MSR[WE], which gates the *doze*, *nap*, and *sleep* power management outputs (internal device signals) from the core complex, is always among the bits saved and restored; hence these outputs negate to the MPC8555E power management logic when the interrupt begins processing in the core. They return to their previous state when the core executes an **rfi**, **rfci**, or **rfmci** instruction. Section 10.1.3, "Interrupts to the Processor Core," lists interrupts that cause the MPC8555E to wake up.

NOTE

Returning *doze*, *nap*, and *sleep* signals to their original state when MSR[WE] is restored differs from how power management is implemented on earlier devices where MSR[POW], which enables power-down requests, is cleared when the processor exits a low-power state and is not automatically restored, as it is in Book E implementations.

18.5.1.8.2 Interrupts and Power Management Controlled by POWMGTCSR

The IRQ_MSK and CI_MSK fields of the POWMGTCSR register prevent *int* interrupts or *cint* critical interrupts from waking the device from a low power state. This is true regardless of the method used to enter the low power state.

Any unmasked interrupt (not masked by the mask bits in the POWMGTCSR register) causes the POWMGTCSR[DOZ,SLP] fields to be cleared when it occurs. When such an interrupt occurs, the device returns to the normal operating mode and does not automatically attempt to return to a low power state after the interrupt is handled.

Note that interrupts caused by the unconditional debug event ($\overline{\text{UDE}}$) and machine check ($\overline{\text{MCP}}$) signals are not masked by the IRQ_MSK and CI_MSK fields; therefore, when these signals assert, the POWMGTCSR[DOZ,SLP] fields are cleared and the device will return to full power operation. See Section 18.4.1.12, "Power Management Control and Status Register (POWMGTCSR)," for detailed information about the bits of POWMGTCSR.

Note also that unmasked interrupts that occur while the device is in the process of going into the sleep state (before sleep is completely attained) can also cause the device to clear the POWMGTCSR[DOZ,SLP] fields and return the device to full power operation.

18.5.1.9 Snooping in Power-Down Modes

When the MPC8555E is in doze mode, the e500 core is in the core-halted state and it snoops its L1 caches and full coherency is maintained. In deeper power-down modes, however, the e500 core does not respond to snoops.

The MPC8555E does not perform dynamic bus snooping as described in the *e500 Reference Manual*. That is, when the e500 core is in the core-stopped state (which is the state of the core when the MPC8555E is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, the L1 caches should be flushed if coherency is required during these power-down modes.

18.5.1.10 Software Considerations for Power Management

Setting MSR[WE] generates a request to the MPC8555E logic (external to the core complex) to enter a power saving state. It is assumed that the desired power-saving state (doze, nap, or sleep) was set up by setting the appropriate HID0 bit, typically at system start-up time. Setting WE has no direct effect on instruction execution, but is reflected on the internal *doze*, *nap*, and *sleep* signals, depending on the HID0 settings. To ensure a clean transition into and out of a power-saving mode, the following program sequence is recommended:

```
            sync
            mtmsr (WE)
            isync
loop:       br loop
```

18.5.1.11 Requirements for Reaching and Recovering from Sleep State

In order to successfully reach the sleep state, I/O traffic to the device must be stopped. The logic that controls the power down sequence waits for all I/O interfaces to become idle. In some applications this may happen eventually without actively shutting down interfaces, but most likely, software will have to take steps to shut down the TSECs, CPM, and PCI interfaces before issuing the command (either the write to the core MSR[WE] as described above or writing to POWMGTCSR) to put the device into sleep state.

The PCI interfaces will begin retrying inbound transactions before entering a power down state. The PCI interfaces, however, could potentially be in an unknown state when they exit sleep if they were in the middle of a retry sequence when internal clocks were shut down. Therefore it is strongly recommended that system software clear the memory space bit in the PCI Bus Command Register before putting the device in sleep mode. Software may also need to set the Agent Config Lock bit of the PCI Bus Function Register so that the device will not respond to configuration transactions. Upon exiting sleep mode, software should return these configuration bits to their normal state.

18.5.2 General-Purpose I/O Signals

Certain groups of signals can optionally be used as general-purpose I/O signals when not being used for their primary function. The general-purpose I/O functionality of these signals can be enabled through configuration registers in the global utilities block. These signals are the following:

- PCI2_AD[15:8] and PCI2_AD[7:0]. When configured as general-purpose I/O, PCI2_AD[15:8] function as outputs and PCI2_AD[7:0] function as inputs.

  PCI2_AD[15:0] can be used as general-purpose I/O if the PCI2 interface is disabled or if the PCI1 controller is disabled or in 32-bit mode.
- TSEC2_RxD[0:7] and TSEC2_TxD[0:7]. TSEC2 pins are fixed as either inputs or outputs based on the direction of the signal's primary function. The TSEC2_TxD pins are always outputs, so

these signals may only be used as outputs when configured as general-purpose I/O. Similarly, the TSEC2_RxD pins are used as inputs when configured as general-purpose I/O.

The TSEC2 TxD and RxD pins are available when the TSEC2 block is disabled. The TxD signals can then be enabled as general-purpose outputs and the RxD pins can be enabled as general-purpose inputs.

When configured as general-purpose I/O signals, software can read inputs by reading the associated GPIO data register. Output values can be set by writing the to the associated GPIO data register. For details regarding the control and status of the general-purpose I/O signals, see Section 18.4.1.7, "General-Purpose I/O Control Register (GPIOCR)."

18.5.3 Interrupt and Local Bus Signal Multiplexing

Except for the CPM, the MPC8555E has very little signal multiplexing. Two sets of DMA channel triggering signals can alternately be placed on other signals as follows:

- $\overline{\text{LCS}}$[5:7] are multiplexed with DMA channel 2 $\overline{\text{DMA\_DREQ2}}$, $\overline{\text{DMA\_DACK2}}$, and $\overline{\text{DMA\_DONE2}}$.
- IRQ[9:11] are multiplexed with DMA channel 3 $\overline{\text{DMA\_DREQ3}}$, $\overline{\text{DMA\_DACK3}}$, and $\overline{\text{DMA\_DDONE3}}$.

For details regarding the selection of the alternate function DMA trigger, see Section 18.4.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

The multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 45, "Parallel I/O Ports," for details on CPM signal multiplexing.

Chapter 19
Performance Monitor

This chapter describes the MPC8555E performance monitor facility, which can be used to monitor and optimize performance. The e500 core implements a separate performance monitor for strictly core-related behavior, such as instruction timing and L1 cache operations. This is described in the *PowerPC™ e500 Core Family Reference Manual* (Freescale Document E500CORERM).

Section 19.4.7, "Performance Monitor Events," briefly describes the events that can be monitored. Refer to the individual chapters for a better understanding of these events.

19.1 Introduction

The MPC8555E includes a performance monitor facility that can be used to monitor and record selected behaviors of the integrated device. Although the performance monitor described here is similar in many respects to the performance monitor facility implemented on the e500 core, it differs in that it is implemented using memory-mapped registers and it counts events outside the e500 core, for example, PCI, DDR, and L2 cache events.

Performance monitor counters (PMC0–PMC8) are used to count events selected by the performance monitor local control registers. PMC0 is a 64-bit counter specifically designated to count cycles. PMC1–PMC8 are 32-bit counters that can monitor 64 counter-specific events in addition to counting 64 reference events.

The benefits of the on-chip performance monitor are numerous, and include the following:

- Because some systems or software environments are not easily characterized by signal traces or benchmarks, the performance monitor can be used to understand the MPC8555E behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.
- System performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

19.1.1 Overview

Figure 19-1 is a high-level block diagram of the performance monitor, which consists of a global control register (PMGC0), one 64-bit counter (PMC0), eight 32-bit counters, and two control registers per counter

(18 total control registers). The global control register PMGC0 affects all counters and takes priority over local control registers. The local control registers are divided into two groups, as follows:

- Local control A registers control counter freezing, overflow condition enable, event selection, and burstiness. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers control the start and stop triggering, contain the counters' threshold values, and the value of the threshold multiplier. Local control register PMLCB0, which controls PMC0, does not contain threshold information because PMC0 only counts cycles.

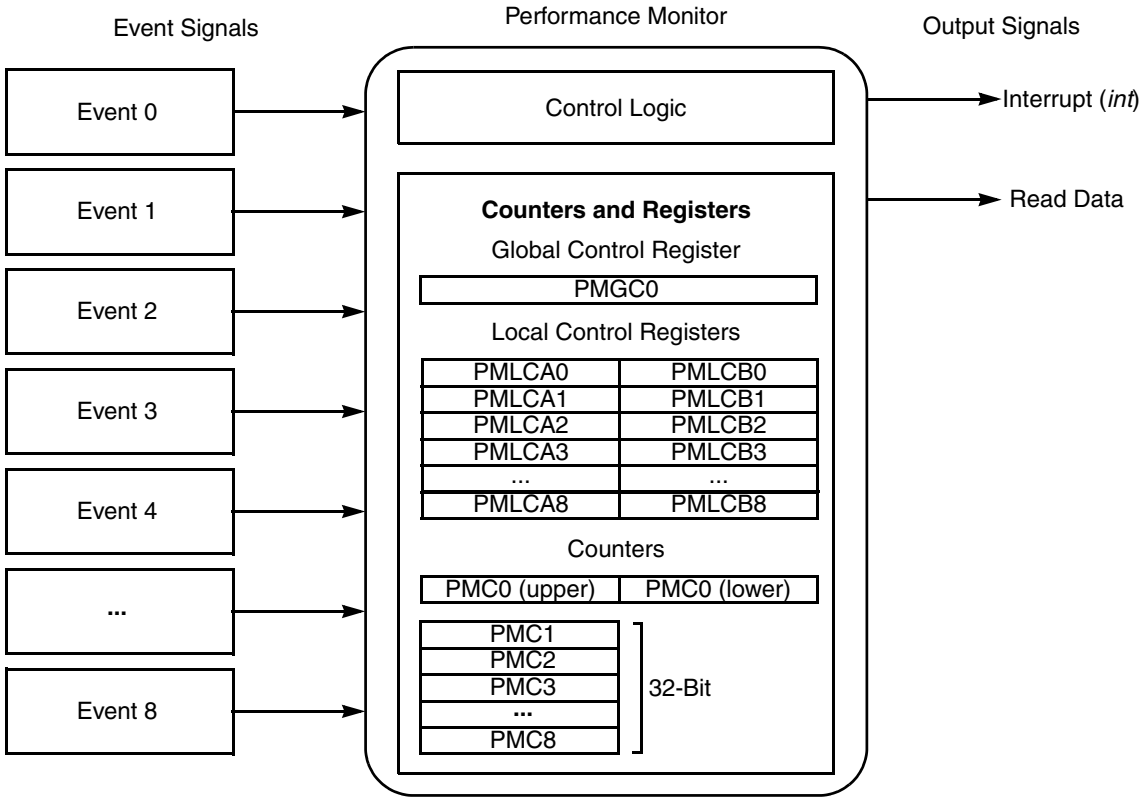


Figure 19-1. Performance Monitor Block Diagram

Performance monitor events are signaled by the functional blocks in the integrated device and are selectively recorded in the PMCs. Sixty-four of these events are referred to as reference events, which can be counted on any of the eight counters. Counter-specific events can be counted only on the counter where the event is defined.

The performance monitor can generate an interrupt on overflow. Several control registers specify how a performance monitor interrupt is signaled. The PMCs can also be programmed to freeze when an interrupt is signaled.

19.1.2 Features

The MPC8555E performance monitor offers a rich set of features that permits a complete performance characterization of the implementation. These features include:

- One 64-bit counter exclusively dedicated to counting cycles
- Eight 32-bit counters that count the occurrence of selected events
- One global control register (affects all counters) and two local control registers per counter
- Ability to count up to 64 reference events that may be counted on any of the eight 32-bit counters
- Ability to count up to 512 counter-specific events
- Triggering and chaining capability
- Duration threshold counting
- Burstiness feature that permits counting of burst events with a programmable time between bursts
- Ability to generate an interrupt on overflow

19.2 Signal Descriptions

The performance monitor does not have any signals that are driven externally (off-chip) but it does assert the internal interrupt (*int*) signal on a performance monitor interrupt condition.

19.3 Memory Map and Register Definition

Performance monitor registers reside in the run-time register block starting at offset 0xE_1000. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved. This section describes the registers implemented to support the performance monitor facilities. Table 19-1 lists the performance monitor registers. These registers can be read or written only with 32-bit accesses.

19.3.1 Register Summary

The performance monitor uses nine counter registers and a group of local control registers that are used to specify the method of counting. Two local control registers are associated with each counter in addition to a global control register that applies to all counters.

Table 19-1. Control Register Memory Map

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	19.3.2.1/19-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1018	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	19.3.3.1/19-9
0xE_101C	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	19.3.2.2/19-5

Table 19-1. Control Register Memory Map (continued)

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	19.3.3.1/19-9

In addition to these registers, the interrupt control provides four pairs of mask registers that can be used to monitor message, interprocessor, timer, and external interrupts. See Section 10.3.4, “Performance Monitor Mask Registers (PMMRs).”

19.3.2 Control Registers

This section describes the performance monitor control registers in detail.

19.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in Figure 19-2, is a 32-bit register used to control all PMCs.

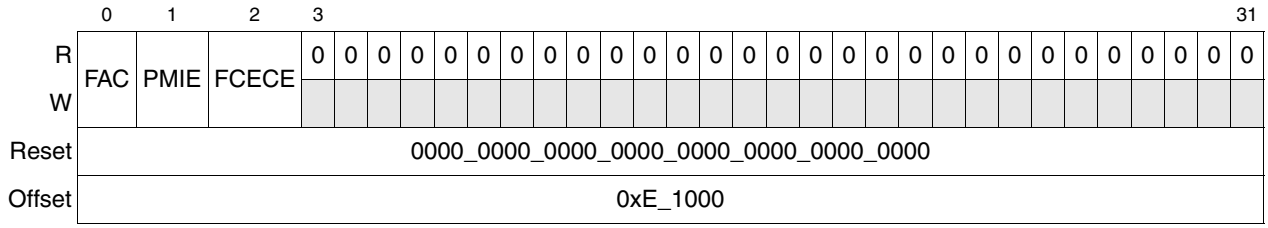


Figure 19-2. Performance Monitor Global Control Register (PMGC0)

Table 19-2 describes PMGC0 fields.

Table 19-2. PMGC0 Field Descriptions

Bits	Name	Description
0	FAC	Freeze all counters. 0 PMCs are incremented (if permitted by other PMGC0/PMLC bits). 1 PMCs are not incremented. Set by hardware when an interrupt is signaled and FCECE = 1.
1	PMIE	Performance monitor interrupt enable. Interrupts are caused by PMC overflows. 0 Interrupts are disabled. 1 Interrupts are enabled and occur when an enabled condition or event occurs.
2	FCECE (DISCOUNT)	Freeze counters on enabled condition or event. An enabled condition or event is defined as: The msb = 1 in PMC*n* and PMLCA*n*[CE] = 1. The use of the trigger and freeze counter conditions depends on the enabled condition. 0 PMCs can be incremented (if permitted by other control bits). 1 PMCs can be incremented (if permitted by other control bits) only until an enabled condition or event occurs, at which time PMGC0[FAC] is set. It is up to software to clear FAC.
3–31	—	Reserved

19.3.2.2 Performance Monitor Local Control Registers (PMLCA*n*, PMLCB*n*)

The performance monitor local control registers (PMLCA*n* and PMLCB*n*) are used to control the operation of the PMCs. The performance monitor local control A and B registers are paired 32-bit control registers that are associated with an individual counter to specify how the counter is used and what event is monitored on that counter.

Figure 19-3 shows the performance monitor local control A0 register (PMLCA0).

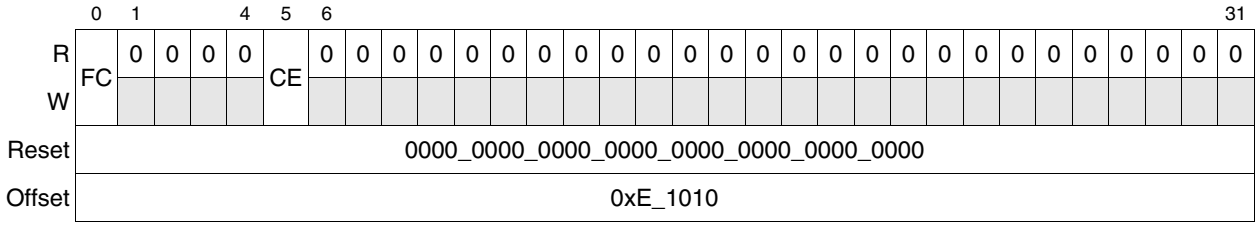


Figure 19-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 19-3 describes PMLCA0 fields.

Table 19-3. PMLCA0 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter. Basic counter enable. 0 The PMCs are enabled and incremented (if permitted by other SPM control bits). 1 The PMCs are disabled–they do not increment.
1–4	—	Reserved
5	CE	Condition enable. Controls counter overflow condition. Should be cleared when PMC0 is used as a trigger or is selected for chaining. 0 Overflow conditions for PMC0 cannot occur (PMC0 cannot cause interrupts or freeze counters) 1 Overflow conditions occur when PMC0[msb] is set.
6–31	—	Reserved

Figure 19-4 shows the performance monitor local control registers A1–A8.

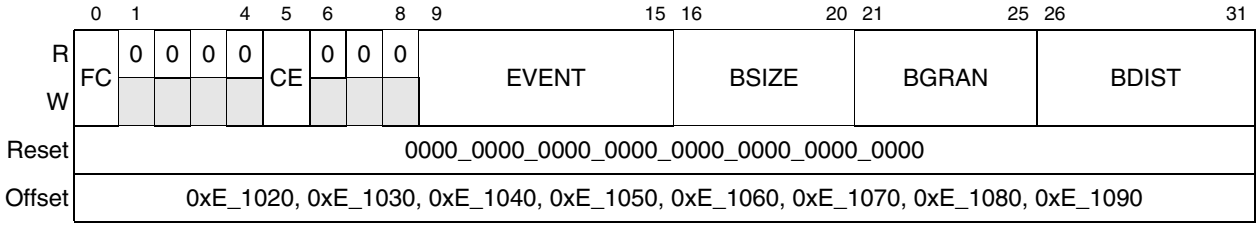


Figure 19-4. Performance Monitor Local Control A Registers (PMLCA1–PMLCA8)

Table 19-4 describes PMLCA*n* fields.

Table 19-4. PMLCA1–PMLCA8 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter 0 The PMCs are incremented (if permitted by other PMC control bits). 1 The PMCs are not incremented (if permitted by other PMC control bits).
1–4	—	Reserved

Table 19-4. PMLCA1–PMLCA8 Field Descriptions (continued)

Bits	Name	Description
5	CE	Condition enable 0 Overflow conditions for PMC*n* cannot occur (PMC*n* cannot cause interrupts or freeze counters). Should be cleared when PMC*n* is used as a trigger or is selected for chaining. 1 Overflow conditions occur when PMC*n*[msb] is set.
6–8	—	Reserved
9–15	EVENT	Event selector. Up to 128 events selectable. See Table 19-10 for definition of events.
16–20	BSIZE	Burst size. Fewest event occurrences that constitute a burst, that is, a rapid sequence of events followed by a relatively long pause. A value less than two implies regular event counting. Any non-threshold, regular event may be counted in a bursty fashion. See Section 19.4.6, “Burstiness Counting,” for more information.
21–25	BGRAN	Burst granularity. The maximum number of clock cycles between events that are considered part of a single burst. See Section 19.4.6, “Burstiness Counting.”
26–31	BDIST	Burst distance (used with TBMULT). The number of clock cycles between bursts. Must be set to a value greater than BSIZE for proper burstiness counting behavior. 00_0000 Regular counting

Figure 19-5 shows the performance monitor local control B0 register (PMLCB0).

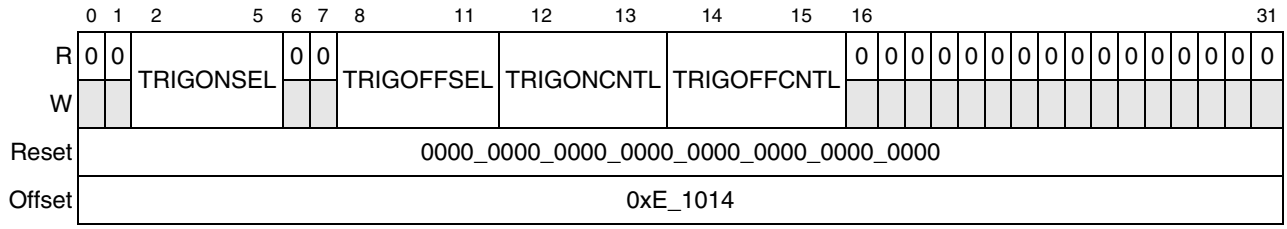


Figure 19-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 19-5 describes PMLCB0 fields.

Table 19-5. PMLCB0 Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. The number of the counter that starts event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. The number of the counter that stops event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved

Table 19-5. PMLCB0 Field Descriptions (continued)

Bits	Name	Description
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–31	—	Reserved

Figure 19-6 shows performance monitor local control registers 1–8.

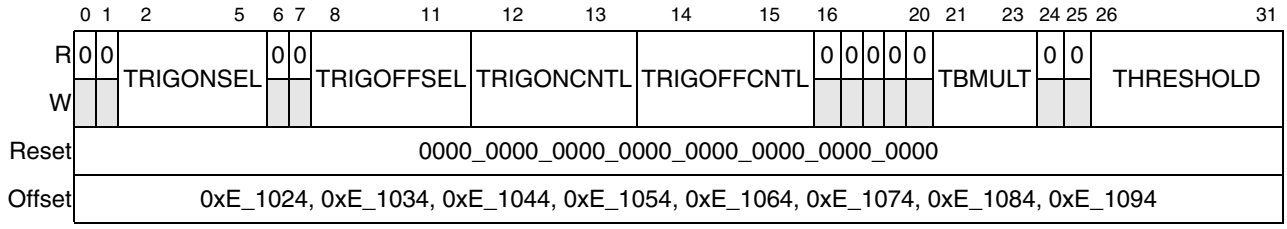


Figure 19-6. Performance Monitor Local Control Register B (PMLCB1–PMLCB8)

Table 19-5 describes PMLCB*n* fields.

Table 19-6. PMLCB1–PMLCB8 Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. Set this field equal to the number of the counter that should trigger event counting to start. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. Set this field equal to the number of the counter that should trigger event counting to stop. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–20	—	Reserved

Table 19-6. PMLCB1–PMLCB8 Field Descriptions (continued)

Bits	Name	Description
21–23	TBMULT	Threshold and burstiness multiplier. Threshold events are counted when the event duration exceeds a specified threshold value. The threshold is scaled based on the TBMULT settings. The burst distance for burstiness counting is also scaled using the TBMULT settings. For all events that scale the threshold, the threshold field is multiplied by the factors shown below (ranging from 1 to 128). 000 1 001 2 010 4 011 8 100 16 101 32 110 64 111 128
24–25	—	Reserved
26–31	THRESHOLD	Threshold. Only events whose (number of) occurrences exceed this value are counted. By varying the threshold value, software can characterize the events subject to the threshold. For example, if PMC2 countsTSEC BD read latencies for which the duration exceeds the threshold, software can obtain the distribution ofTSEC BD read latencies for a given program by monitoring the program repeatedly using a different threshold value each time.

19.3.3 Counter Registers

This section describes the PMCs in detail.

NOTE

Because accessing a PMC manually has priority over incrementing it due to event counting, reading or writing a PMC while it is counting may affect the count. Likewise, accessing a performance monitor control register while its target counter is counting may also affect the count.

19.3.3.1 Performance Monitor Counters (PMC0–PMC8)

PMC0–PMC8 are used to count events selected by the performance monitor local control registers. PMC0, shown in Figure 19-7, is associated with two 32-bit registers that form a 64-bit counter designated to count clock cycles. PMC0 upper represents the upper 32 bits of counter 0, and PMC0 lower represents the lower 32 bits.

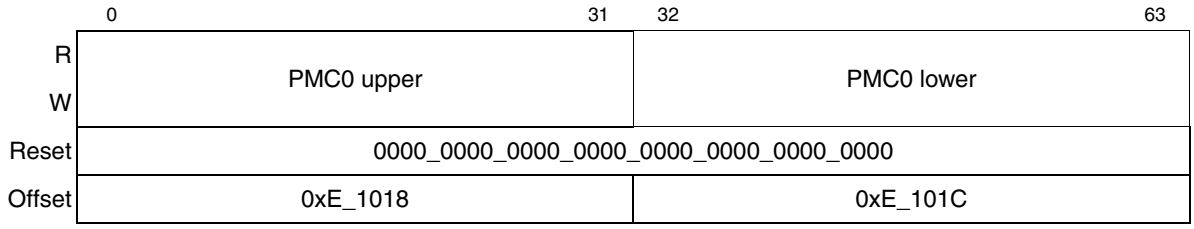


Figure 19-7. Performance Monitor Counter Register 0 (PMC0)

Table 19-7 describes PMC0 fields.

Table 19-7. PMC0 Field Descriptions

Bits	Name	Description
0–63	PMC0	Event count. Counts only clock cycles

PMC1–PMC8, shown in Figure 19-8, are 32-bit counters that can monitor 64 unique events in addition to the 64 reference events that can be counted on all of these registers.

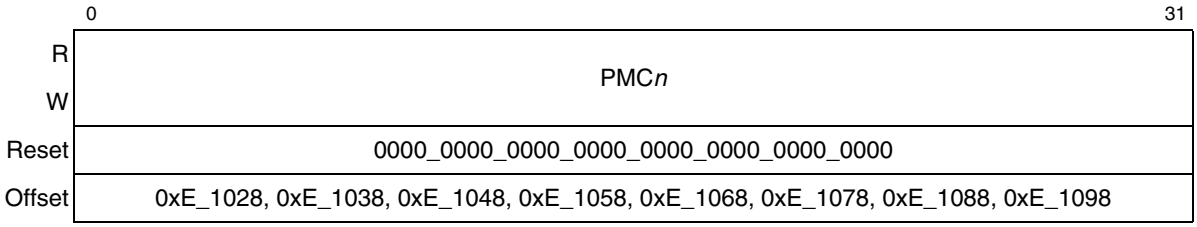


Figure 19-8. Performance Monitor Counter Register (PMC1–PMC8)

Table 19-8 describes PMC*n* fields.

Table 19-8. PMC1–PMC8 Field Descriptions

Bits	Name	Description
0–31	PMC*n*	Event count. An overflow is indicated when the msb = 1. Manually setting the msb can cause an immediate interrupt.

19.4 Functional Description

This section describes the use of some features of the performance monitor.

19.4.1 Performance Monitor Interrupt

PMCs can generate an interrupt on an overflow when the msb of a counter changes from 0 to 1. For the interrupt to be signaled, the condition enable bit (PMLCA*n*[CE]) and performance monitor interrupt enable bit (PMGC0[PMIE]) must be set. When an interrupt is signaled and the freeze-counters-on-enabled-condition-or-event bit (PMGC0[FCECE]) is set, PMGC0[FAC] is set by hardware and all of the registers are frozen. Software can clear the interrupt condition by resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

19.4.2 Event Counting

Using the control registers described in Section 19.3.2, “Control Registers,” the nine PMCs can count the occurrences of specific events. The 64-bit PMC0 is designated to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC8). Additionally, up to 64 unique events can be counted on each 32-bit counter.

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers.þCounters can be frozen

individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

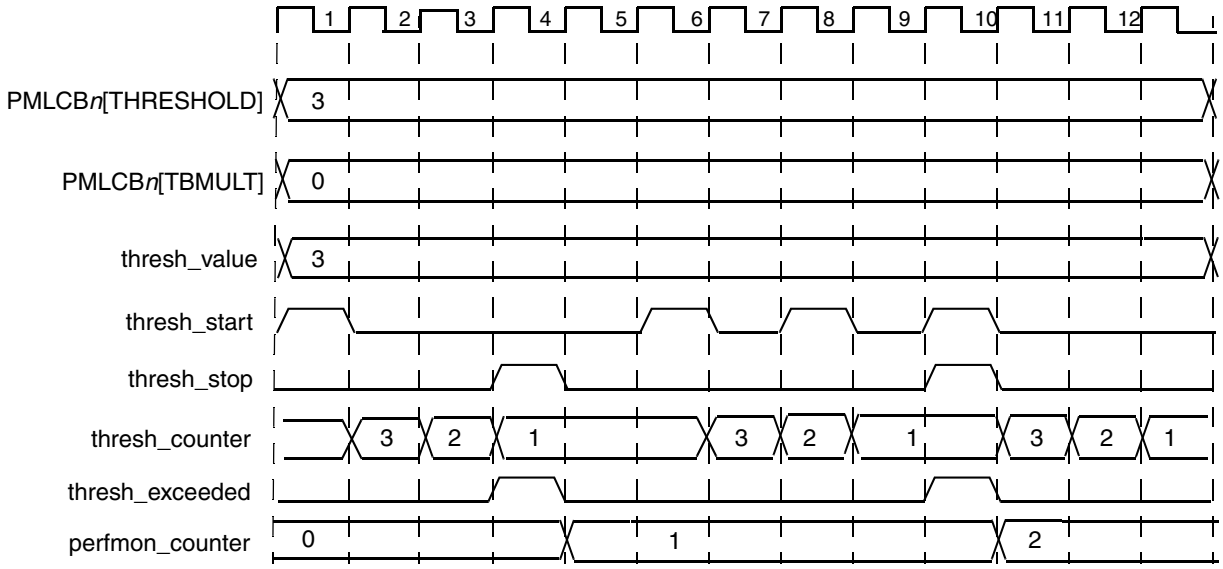
19.4.3 Threshold Events

The threshold feature allows characterization of events that can take a variable number of clock cycles to occur. Threshold events are counted only if the latency is greater than the threshold value specified in PMLCB*n*[THRESHOLD].

For duration threshold event sequences, the PMC increments only when the duration of the event is equal to or greater than the threshold value. The threshold value is scaled by a multiple specified in PMLCB*n*[TBMULT].

A threshold event requires two signals: The first indicates when a threshold event sequence begins, and the second indicates when it ends. An internal counter determines when the threshold count is exceeded and when the PMC can increment. This internal counter decrements during a threshold event sequence until it reaches the value of one. A new sequence cannot begin until the current one completes. Additional threshold start signals are ignored during a sequence until a threshold stop signal occurs. If both a start and stop signal are asserted during the same cycle in a current sequence, the stop terminates the current sequence and the start signals the beginning of a new one. However, if both signals are asserted during the same cycle while not in a current event sequence, both signals are ignored. Figure 19-9 is a timing diagram for duration threshold event counting.

An illegal condition exists if the threshold value obtained from PMLCB*n*[THRESHOLD] and PMLCB*n*[TBMULT] is less than two. Under these conditions the intent of threshold counting is ambiguous.



1 For this example a threshold value of three indicates that the user wishes to count the number of times a particular event lasts three cycles or longer.

Figure 19-9. Duration Threshold Event Sequence Timing Diagram

19.4.4 Chaining

By configuring one counter to increment each time another counter overflows, several counters can be chained together to provide event counts larger than 32 bits. Each counter in a chain adds 32 bits to the maximum count. The register chaining sequence is not arbitrary and is specified indirectly by selecting the register overflow event to be counted. Selecting an event has the effect of selecting a source register because all available chaining events, as shown in Table 19-10, are dedicated to specific registers.

Note that the chaining overflow event occurs when the counter reaches its maximum value and wraps, not when the register's msb is set. For this overflow to occur, PMLCA*n*[CE] should be cleared to avoid signaling an interrupt when the counter's most significant bit is set. Note that several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

19.4.5 Triggering

Triggering allows one counter to start or stop counting on the change of another counter or on the overflow of another counter. More specifically, if PMC1 is set to start or stop counting as a result of a change or overflow in counter PMC2, then counter PMC2 must be identified in the local control register of counter PMC1. This is done by appropriately setting the trigger-on select bit or trigger-off select bit (PMLCB1[TRIGOFFSEL] or PMLCB1[TRIGONSEL]). Additionally, the condition that triggers the counter must be selected by configuring the corresponding control bits (PMLCB1[TRIGONCNTL] or PMLCB1[TRIGOFFCNTL]). Assuming the counter is enabled by other control register settings, the counter increments (or freezes) when its specified event occurs after the trigger-on (or off) condition occurs.

When trigger on and trigger off are both selected, the trigger-off condition is ignored until the trigger-on condition has occurred. Furthermore, when a trigger-off condition occurs, the counter state is preserved; it is not restarted by subsequent trigger-on conditions.

Triggering is disabled when the counter's trigger-select bits specify itself as the trigger source. Similarly, triggering is disabled when the trigger control bits are cleared.

19.4.6 Burstiness Counting

The burstiness counting feature makes it easier to characterize events that occur in rapid succession followed by a relatively long pause. As shown in Table 19-9, event bursts are defined by size, granularity, and distance.

Table 19-9. Burst Definition

Parameter	Description	Register Field
Size	The minimum number of events constituting a burst	PMLCA*n* [BSIZE]
Granularity	The maximum time between individual events counted as members of the same burst	PMLCA*n* [BGRAN]
Distance	The minimum time between bursts	PMLCA*n* [BDIST] x PMLCB*n*[TBMULT]

Figure 19-10 shows the relationships between size, granularity, and distance. Burstiness counting can be performed for all events except threshold events.

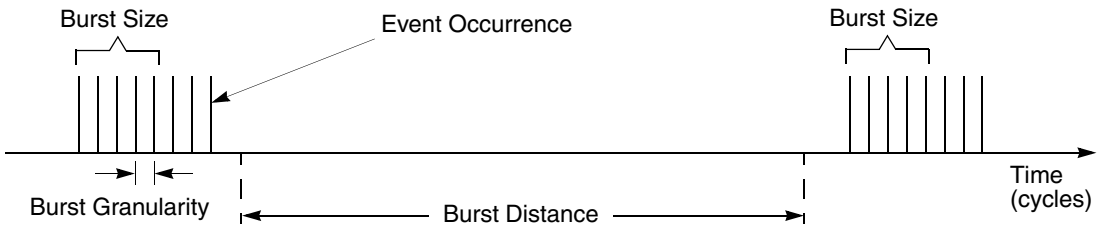


Figure 19-10. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCA*n*[BSIZE]) specifies the minimum number of event occurrences that constitute a burst. A burst is identified when the number of event occurrences equals or exceeds PMLCA*n*[BSIZE]. Furthermore, these individual event occurrences must be separated by no more clock cycles than the value in the burstiness granularity field (PMLCA*n*[BGRAN]). Note that, although a burst is identified when the minimum number of events occurs, it is not counted until the burst sequence has ended. A burst sequence ends when the specified burstiness granularity is exceeded, at which point the last valid event has occurred for that sequence.

PMLCA*n*[BGRAN] specifies the maximum number of cycles between individual events for them to qualify as members of the same burst sequence.

The burstiness distance field (PMLCA*n*[BDIST]) and threshold/burstiness multiplier field (PMLCB*n*[TBMULT]) specify the acceptable number of cycles between the end of a burst sequence and the beginning of a new sequence for a group of event occurrences to be counted as an individual burst. The product of the burstiness distance field and the threshold/burstiness multiplier field determine the

burstiness distance value used to determine when another burst sequence can begin. Note that the burst distance count begins when a new burst sequence ends and the PMC is incremented. No new burst sequence may begin until the burst distance count has reached zero. After the burst distance count reaches zero, it holds the zero value indicating that a new burst sequence can be counted. The burst distance count begins again when a new burst sequence is identified and counted.

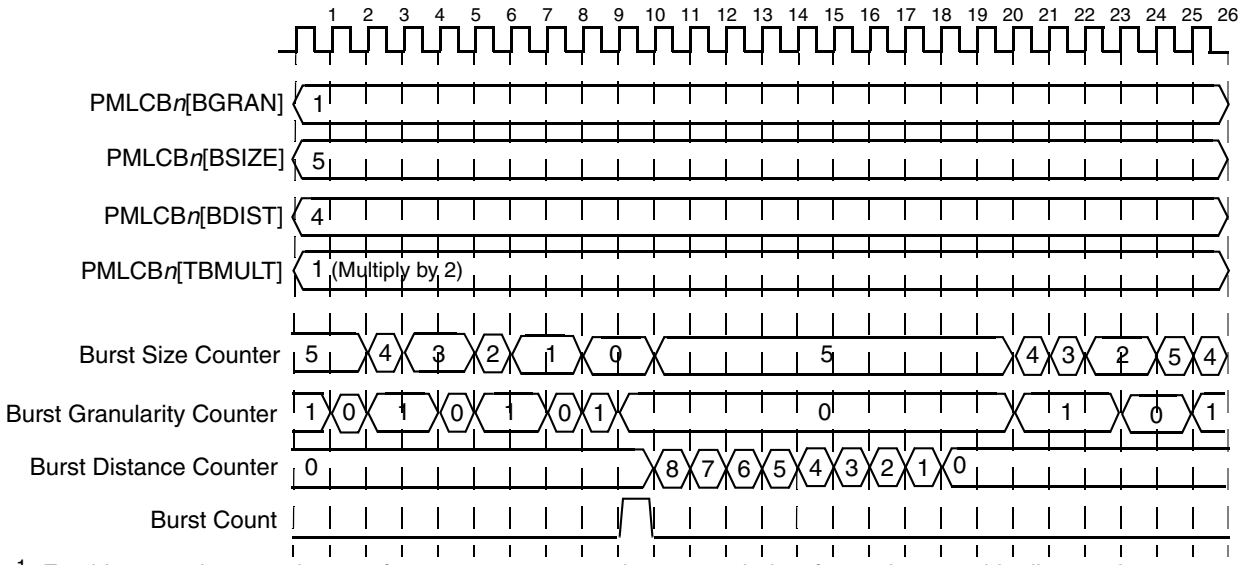
Burstiness counting is disabled when the definition of a burst is ambiguous, that is, when the burst size field is less than two, or the burst distance is zero. When burstiness counting is disabled, regular counting is allowed.

Figure 19-10 shows that the burst distance is measured from the end of one burst sequence and that a new burst sequence may not begin until the burst distance count expires.

Three internal counters track the different values required for burstiness counting.

- Burstiness size is monitored by a counter. It is loaded with the value specified in the local control register when the burst granularity counter and the burst distance counters reach zero, and no new event is occurring. It always decrements when the following conditions occur: its value is not already zero, an event occurs, and the burst distance count equals zero.
- Burstiness granularity is monitored by a counter that is loaded with the specified value in the local control register on the rising edge of an event occurrence whenever the burst distance count equals zero. The granularity counter is decremented (if it has not already reached zero) when an event is not occurring and burst distance count equals zero.
- Burstiness distance is measured by a counter that is loaded with the product of PMLCB*n*[BDIST] and PMLCB*n*[TBMULT] when a burst sequence has been identified and counted. This counter is decremented when burstiness counting is enabled (and the counter has not already reached zero).

A burst is counted at the end of a burst sequence when the three burst parameter counters are all equal to zero. Figure 19-11 shows a burstiness counting example.



1 For this example: count bursts of 5 event occurrences, burst granularity of 1, and acceptable distance between bursts of 8 (product of TBMULT and BDIST).

Figure 19-11. Burstiness Counting Timing Diagram

19.4.7 Performance Monitor Events

Table 19-10 lists performance monitor events specified in PMLCA1–PMLC8.

The event assignment column indicates the event's type and number, using the following formats:

- Ref:#—Reference events are shared across counters PMC1–PMC8. The number indicates the event. For example, Ref:6 means that PMC1–PMC8 share reference event 6.
- C[0–8]:#—Counter-specific events. C8 indicates an event assigned to PMC8. Thus C8:62 means PMC8 is assigned event 62 (PIC interrupt wait cycles).

  Note that with counter-specific events, an offset of 64 must be used when programming the field, because counter-specific events occupy the bottom 64 values of the 7-bit event field where events are numbered. For example, to specify counter-specific event 0, the event field must be programmed to 64.

Counter events not specified in Table 19-10 are reserved.

Table 19-10. Performance Monitor Events

Event Counted	Number	Description of Event Counted
General Events		
Nothing	Ref:0	Register counter holds current value
System cycles	C0	CCB (platform) clock cycles
DDR Memory Controller Events		
Cycles a read is returning data from DDR SDRAM	Ref:10	Each data beat returned to the memory controller on the DDR SDRAM interface
Cycles a read or write transfers data from (or to) DDR SDRAM	Ref:11	Each data beat transferred to or from the DDR SDRAM
Pipelined read misses in the row open table	C1:57	Row open table read misses issued while a read is outstanding
Pipelined read or write misses in the row open table	C2:0	Row open table read or write misses issued while a read or write is outstanding
Non-pipelined read misses in the row open table	C3:60	Row open table read misses issued when no reads are outstanding
Non-pipelined read or write misses in the row open table	C4:0	Row open table read or write misses issued when no reads or writes are outstanding
Pipelined read hits in the row open table	C5:56	Row open table read hits issued when a read is outstanding
Pipelined read or write hits in the row open table	C6:0	Row open table read or write hits issued when a read or write is outstanding
Non-pipelined read hits in the row open table	C7:57	Row open table read hits issued when no reads are outstanding
Non-pipelined read or write hits in the row open table	C8:0	Row open table read or write hits issued when no reads or writes are outstanding

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Forced page closings not caused by a refresh	C1:0	Precharges issued to the DDR SDRAM for any reason except refresh. The possibilities are as follows: • A new transaction must be issued to an already active bank and sub-bank that has a different row open. • A new transaction must be issued, but the row open table is full and there is no bank/sub-bank match between the current transaction and the row open table. • The BSTOPRE interval expired for an open row.
Row open table misses	C2:1	Transactions that miss in the row open table
Row open table hits	C3:0	Transaction that hit in the row open table
Force page closings	C4:1	Forced page closings including those due to refreshes
Read-modify-write transactions due to ECC	C5:0	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence is issued on the DDR SDRAM interface.
Forced page closings due to collision with bank and sub-bank	Ref:12	Increments if a new transaction must be issued to an active bank and sub-bank that has a different row open
Reads or writes from core	Ref:13	—
Reads or writes from TSEC 1	C3:1	—
Reads or writes from TSEC 2	C4:2	—
Reads or writes from CPM	C5:1	—
Reads or writes from PCI	C4:3	—
Reads or writes from DMA	C5:2	—
Row open table hits for reads or writes from core	Ref:14	—
Row open table hits for reads or writes from TSEC 1	C6:1	—
Row open table hits for reads or writes from TSEC 2	C7:0	—
Row open table hits for reads or writes from CPM	C8:1	—
Row open table hits for reads or writes from PCI	C7:1	—
Row open table hits for reads or writes from DMA	C8:2	—
Memory Target Queue Events		
MEM TQ read/write address collision	C5:5	—
Misaligned engine priority 2 occupied	C5:12	Misaligned engine for priority 2 transactions busy
Misaligned engine priority 1 occupied	C6:12	—
Misaligned engine priority 0 occupied	C7:10	Misaligned engine for priority 0 transactions busy

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ACK history queue full	C4:61	Unacked packets are outstanding.
DMA Controller Events		
Channel 0 read request	C1:2	DMA channel 0 read request active in the system
Channel 1 read request	C2:5	DMA channel 1 read request active in the system
Channel 2 read request	C3:4	DMA channel 2 read request active in the system
Channel 3 read request	C4:6	DMA channel 3 read request active in the system
Channel 0 write request	C1:3	DMA channel 0 write request active in the system
Channel 1 write request	C2:6	DMA channel 1 write request active in the system
Channel 2 write request	C3:5	DMA channel 2 write request active in the system
Channel 3 write request	C4:7	DMA channel 3 write request active in the system
Channel 0 descriptor request	C5:41	DMA channel 0 descriptor request active in the system
Channel 1 descriptor request	C6:44	DMA channel 1 descriptor request active in the system
Channel 2 descriptor request	C7:41	DMA channel 2 descriptor request active in the system
Channel 3 descriptor request	C8:41	DMA channel 3 descriptor request active in the system
Channel 0 read DW or less	C1:4 and C5:53	DMA channel 0 read double word valid
Channel 1 read DW or less	C2:7 and C6:58	DMA channel 1 read double word valid
Channel 2 read DW or less	C3:6 and C7:54	DMA channel 2 read double word valid
Channel 3 read DW or less	C4:8 and C8:52	DMA channel 3 read double word valid
Channel 0 write DW or less	C1:5	DMA channel 0 write double word valid
Channel 1 write DW or less	C2:8	DMA channel 1 write double word valid
Channel 2 write DW or less	C3:7	DMA channel 2 write double word valid
Channel 3 write DW or less	C4:9	DMA channel 3 write double word valid
e500 Coherency Module (ECM) Events		
ECM request wait core	C8:13	Asserted for every cycle core request occurs
ECM request wait CPM/SAP/boot sequencer	C7:13	Asserted for every cycle CPM request occurs
ECM request wait TSEC1	C5:16	Asserted for every cycle TSEC1 request occurs
ECM request wait TSEC2	C6:16	Asserted for every cycle TSEC2 request occurs
ECM request wait PCI/DMA	C4:20	Asserted for every cycle PCI request occurs
ECM dispatch	Ref:15	ECM dispatch (includes address only's) Note: all ECM dispatch events are for committed dispatches
ECM dispatch from core	C1:16	ECM dispatch from core (includes address only's)
ECM dispatch from CPM	C2:20	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ECM dispatch from TSEC1	C3:19	—
ECM dispatch from TSEC2	C4:21	—
ECM dispatch from PCI	C6:17	—
ECM dispatch from DMA	C7:14	—
ECM dispatch from other	C8:14	—
ECM dispatch to DDR	C4:22	—
ECM dispatch to L2/SRAM	C5:18	—
ECM dispatch to LBC	C6:18	—
ECM dispatch to PCI	C8:15	—
ECM dispatch snoopable	C3:20	—
ECM dispatch write	C1:17	—
ECM dispatch write allocate	C2:21	—
ECM dispatch write allocate lock	C3:21	—
ECM dispatch read	C4:23	—
ECM dispatch read unlock	C5:19	—
ECM dispatch read clear atomic	C6:19	—
ECM dispatch read set atomic	C7:16	—
ECM dispatch read decrement atomic	C8:16	—
ECM dispatch read increment atomic	C7:17	—
ECM data bus grant DDR	C1:18	—
ECM data bus grant LBC	C2:22	—
ECM data bus grant PIC	C1:19	—
ECM data bus grant CPM	C2:23	—
ECM data bus grant TSEC1	C3:23	—
ECM data bus grant TSEC2	C4:25	—
ECM data bus wait DDR	C5:20	—
ECM data bus wait LBC	C6:20	—
ECM data bus wait PIC	C5:21	—
ECM data bus wait CPM	C6:21	—
ECM data bus wait TSEC1	C7:19	—
ECM data bus wait TSEC2	C8:18	—
ECM global data bus beat	Ref:16	—
ECM e500 direct read bus beat	Ref:17	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ECM e500 direct read bus beat forwarded	C2:24	ECM direct read bus beat forwarded directly to e500 R1 data bus
ECM cancel	Ref:18	—
Interrupt Controller (PIC) Events		
PIC total interrupt count	Ref:26	Total number of interrupts serviced
PIC interrupt wait cycles	C8:62	Counts cycles when an interrupt waits to be acknowledge
PIC interrupt service cycles	C2:19	Number of cycles there is an interrupt currently being serviced.
PIC interrupt select 0 (duration threshold)	C1:56	THRESHOLD: select 0–3: interrupt count over threshold. (Note: only unmasked, nonzero priority requests are acknowledged). The four interrupts are selected through register pairs, PM0MR*n*–PM3MR*n*. See Section 10.3.4, “Performance Monitor Mask Registers (PMMRs).”
PIC interrupt select 1 (duration threshold)	C3:59	
PIC interrupt select 2 (duration threshold)	C5:55	
PIC interrupt select 3 (duration threshold)	C6:60	
PCI Common Events		
PCI clock cycles	Ref:28	—
PCI inbound memory reads	C1:62	Includes all read types.
PCI inbound memory writes	C2:37	—
PCI inbound config reads	C3:63	—
PCI inbound config writes	C4:37	—
PCI outbound memory reads	C5:30	Includes all read types.
PCI outbound memory writes	C6:32	Number of PCI outbound memory writes.
PCI outbound I/O reads	C3:37	—
PCI outbound I/O writes	C4:38	—
PCI outbound config reads	C7:26	Number of PCI outbound config reads.
PCI outbound config writes	C8:26	—
PCI inbound total read data beats	C5:32	Includes 32- and 64-bit transactions.
PCI inbound total write data beats	C6:34	Includes 32- and 64-bit transactions.
PCI outbound total read data beats	C7:28	Includes 32- and 64-bit transactions.
PCI outbound total write data beats	C8:28	Includes 32- and 64-bit transactions.
PCI inbound 32-bit read data beats	C1:30	—
PCI inbound 32-bit write data beats	C2:38	—
PCI outbound 32-bit read data beats	C3:38	—
PCI outbound 32-bit write data beats	C4:39	—
PCI inbound 64-bit read data beats	C5:31	—
PCI inbound 64-bit write data beats	C6:33	—
PCI outbound 64-bit read data beats	C7:27	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PCI outbound 64-bit write data beats	C8:27	—
PCI total transactions	C7:29	Includes 32- and 64-bit transactions.
PCI 64-bit transactions	C8:29	—
PCI inbound purgeable reads	C2:2	—
PCI inbound (speculative reads) purgeable reads discarded	C8:63	
PCI idle cycles	C1:31	—
PCI dual address cycles	C2:40	—
PCI internal cycles	C3:39	—
PCI inbound memory read	C1:34	—
PCI inbound memory readline	C2:44	—
PCI inbound memory read multiple	C3:42	—
PCI outbound memory reads	C4:43	Number of PCI outbound memory reads.
PCI outbound memory read lines	C5:36	Number of PCI outbound memory read lines.
PCI wait	C1:35	$\overline{\text{PCI}n\text{\_IRDY}}$, $\overline{\text{PCI}n\text{\_TRDY}}$ not both asserted
PCI Specific Events		
PCI cycles $\overline{\text{PCI}n\text{\_IRDY}}$ is asserted	C6:36	—
PCI cycles $\overline{\text{PCI}n\text{\_TRDY}}$ is asserted	C7:31	—
PCI cycles $\overline{\text{PCI}n\text{\_FRAME}}$ is asserted	C8:31	—
PCI snoopable	C1:32	—
PCI write stash	C2:42	—
PCI write stash with lock	C3:41	—
PCI read unlock	C4:42	—
PCI byte enable transactions	C1:33	—
PCI non-byte enable transactions	C2:43	—
Three-Speed Ethernet Controller (TSEC)		
TSEC1 Address Data Filtering (ADF) Events		
Accepted frames	Ref:36	—
Individual hash table accepted frame	C7:35	—
Group hash table accepted frame	C8:35	—
Rejected frames	Ref:39	—
Dropped frames	Ref:38	Dropped frames that could have been accepted (data overflow, status overflow and lack of BDs)

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Dropped frames due to data overflow	C1:41	Frames dropped because of overflow in the FIFO and 256-byte buffer
Dropped frames due to status overflow	C2:50	Frames dropped because of inability to write status for one frame and a new frame starts
TSEC1 FIFO Events		
Receive FIFO data valid	C1:45	—
Transmit frames without threshold	C7:44	Transmit frames that don't hit Tx FIFO threshold
Receive FIFO above 1/4	Ref:47	—
Receive FIFO above 1/2	Ref:48	—
Receive FIFO above 3/4	Ref:49	—
TSEC1 DMA Events		
DMA reads	C1:47	Descriptor and data reads
BD reads	C2:54	TxBD and RxBD reads
RxBD reads	C3:51	RxBD reads (transmit number can be calculated by subtracting this number from total)
DMA writes	C4:52	Writes (descriptor and data)
BD writes	C5:46	TxBDs and RxBDs closed
RxBD writes	C6:49	RxBDs closed (transmit number can be calculated by subtracting this number from total)
RxBD read latency (duration threshold)	Ref:41	Times RxBD read latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received
TxBD read latency (duration threshold)	Ref:42	Times TxBD read latency exceeds threshold (threshold event). Start: request asserted Stop: data acknowledge received
RxBD write latency (duration threshold)	Ref:43	Times RxBD write latency exceeds threshold (threshold event). Only for writes that require a response. Start: request asserted Stop: end of transaction received
TxBD write latency (duration threshold)	Ref:44	Times TxBD write response latency exceeds threshold (threshold event). Only for writes that require a response. Start: request asserted Stop: end of transaction received
Tx data read latency (duration threshold)	Ref:45	Times data read response latency exceeds threshold (threshold event). Start: request asserted Stop: data acknowledge received
Data beats	C7:48	—
Read data beats	C8:46	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Rx frame interrupts	C1:49	Times receive frame interrupt is set. Event occurs when the BD receive interrupt is set on a last BD.
Tx frame interrupts	C3:53	Times a transmit frame interrupt is set. Event occurs when the BD transmit interrupt is set on a BD with L = 1.
Rx frame processing (duration threshold)	Ref:46	Times frame processing exceeds threshold (threshold event). Start: open first BD Stop: close last BD
TSEC2 Address Data Filtering (ADF) Events		
Accepted frames	Ref:50	—
Individual hash table accepted frame	C7:36	—
Group hash table accepted frame	C8:36	—
Rejected frames	Ref:53	—
Dropped frames	Ref:52	Dropped frames that could have been accepted (data overflow, status overflow and acknowledgement of BDs)
Dropped frames due to data overflow	C1:42	Frames dropped because of overflow in the FIFO and 256-byte buffer
Dropped frames due to status overflow	C2:51	Frames dropped because of inability to write status for one frame and a new frame starts
TSEC2 FIFO Events		
Receive FIFO data valid	C1:46	Receive FIFO contains receive data
Transmit frames without threshold	C7:45	Transmit frames that don't hit Tx FIFO threshold
Receive FIFO above 1/4	Ref:61	—
Receive FIFO above 1/2	Ref:62	—
Receive FIFO above 3/4	Ref:63	—
TSEC2 DMA Events		
DMA reads	C1:48	Descriptor and data reads
BD reads	C2:55	Transmit and RxBD reads
RxBD reads	C3:52	Times that RxBD reads (transmit number can be calculated by subtracting this number from total)
DMA writes	C4:53	Descriptor and data writes
BD writes	C5:47	Times that TxBDs and RxBDs closed
RxBD writes	C6:50	Times RxBDs closed (transmit number can be calculated by subtracting this number from total)
RxBD read latency (duration threshold)	Ref:55	Times RxBD read latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
TxBD read latency (duration threshold)	Ref:56	Times TxBD read latency exceeds threshold (threshold event). Start: request asserted Stop: data acknowledge received
RxBD write latency (duration threshold)	Ref:57	Times RxBD write response latency exceeds threshold (threshold event). Only for writes that require a response. Start: request asserted Stop: End of Transaction received
TxBD write latency (duration threshold)	Ref:58	Times that TxBD write response latency exceeds threshold (threshold event). Only for writes that require a response. Start: request asserted Stop: End of Transaction received
Tx data read latency (duration threshold)	Ref:59	Data read response latency exceeds threshold (threshold event). Start: request asserted Stop: Data acknowledge received
Data beats	C7:49	—
Read data beats	C8:47	—
Rx frame interrupts	C1:50	BD receive interrupt is set on a last BD.
Tx frame interrupts	C3:54	BD transmit interrupt is set on a BD with L = 1.
Rx frame processing	Ref:60	Receive frame processing exceeds threshold (threshold event). Start: Open first BD Stop: Close last BD
Local Bus Events		
Bank 1 hits (chip-select)	C1:51	—
Bank 2 hits (chip-select)	C2:56	—
Bank 3 hits (chip-select)	C3:55	—
Bank 4 hits (chip-select)	C4:54	—
Bank 5 hits (chip-select)	C5:48	—
Bank 6 hits (chip-select)	C6:53	—
Bank 7 hits (chip-select)	C7:50	—
Bank 8 hits (chip-select)	C8:50	—
Requests granted to CPM port	C1:52	—
Requests granted to ECM port	C2:57	—
Cycles atomic lock for CPM port is enabled	C3:56	—
Cycles atomic reservation for ECM port is enabled	C4:55	
Atomic reservation time-outs for CPM port	C5:49	—
Atomic reservation time-outs for ECM port	C6:54	—
Cycles a read is taking in GPCM	C1:53	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Cycles a read is taking in UPM	C2:58	—
Cycles a read is taking in SDRAM	C3:57	—
Cycles a write is taking in GPCM	C4:56	—
Cycles a write is taking in UPM	C5:50	—
Cycles a write is taking in SDRAM	C6:55	—
SDRAM bank misses	C7:51	—
SDRAM page misses	C8:51	—
L2 Cache/SRAM Events		
Core instruction accesses to L2 that hit	Ref:22	—
Core instruction accesses to L2 that miss	C2:59	—
Core data accesses to L2 that hit	Ref:23	—
Core data accesses to L2 that miss	C4:57	—
Non-core burst write to L2 (cache external write or SRAM)	C5:51	—
Non-core non-burst write to L2	C6:56	—
Noncore write misses cache external write window and SRAM memory range	C7:52	—
Non-core read hit in L2	Ref:24	—
Non-core read miss in L2	C1:54	—
L2 allocates, from any source	Ref:25	—
L2 retries due to full write queue	C2:60	—
L2 retries due to address collision	C3:58	—
L2 failed lock attempts due to full set	C4:58	—
L2 victimizations of valid lines	C5:52	—
L2 invalidations of lines	C6:57	—
L2 clearing of locks	C7:53	—
Debug Events		
External event	C3:61	Number of cycles trig_in pin is asserted
Watchpoint monitor hits	C2:61	—
Trace buffer hits	C1:58	—
DUART Events		
UART0 baud rate	C1:63	—
UART1 baud rate	C5:63	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Chaining Events		
PMC0 carry-out	Ref:1	PMC0[0] 1-to-0 transitions.
PMC1 carry-out	Ref:2	PMC1[0] 1-to-0 transitions. Reserved for PMC1.
PMC2 carry-out	Ref:3	PMC2[0] 1-to-0 transitions. Reserved for PMC2.
PMC3 carry-out	Ref:4	PMC3[0] 1-to-0 transitions. Reserved for PMC3.
PMC4 carry-out	Ref:5	PMC4[0] 1-to-0 transitions. Reserved for PMC4.
PMC5 carry-out	Ref:6	PMC5[0] 1-to-0 transitions. Reserved for PMC5.
PMC6 carry-out	Ref:7	PMC6[0] 1-to-0 transitions. Reserved for PMC6.
PMC7 carry-out	Ref:8	PMC7[0] 1-to-0 transitions. Reserved for PMC7.
PMC8 carry-out	Ref:9	PMC8[0] 1-to-0 transitions. Reserved for PMC8.

19.4.8 Performance Monitor Examples

Table 19-12 contains sample register settings for the four supported modes.

- Simple event performance monitoring example
- Triggering event performance monitoring example
- Threshold event performance monitoring example
- Burstiness event performance monitoring example

The settings in Table 19-11 are identical for all four examples.

Table 19-11. PMGC0 and PMLCA*n* Settings

Field	Setting	Reason
PMGC0[FAC]	0	Counters must not be frozen.
PMGC0[PMIE]	1	Performance monitor interrupts are enabled
PMGC0[FCECE]	1	Counters should be frozen when an interrupt is signaled.
PMLCA*n*[FC]	0	Counters cannot be frozen for counting.
PMLCA*n*[CE]	1	Overflow condition enable is required to allow interrupt signaling.

For simple event counting, a non-threshold event is selected in PMLCA*n*[EVENT] and all other features are disabled by clearing all register fields except for CE.

For the triggering example any event can be selected in PMLCA*n*[EVENT]. All other features are disabled by clearing these register fields except for CE to allow interrupt signaling. If PMLCB*n*[TRIGONSEL] is 3 and PMLCB*n*[TRIGOFFSEL] is 5, the counter begins and ends counting based on the conditions in counters three and five. Furthermore, if PMLCB*n*[TRIGONCNTL] is 1, the counter begins counting when PMC3 changes value. According to the setting in PMLCB*n*[TRIGOFFCNTL], the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, PMLCA*n*[CE] for this

counter must be cleared so that interrupt signaling is not enabled and the counter does not freeze when it overflows.

For threshold counting, a threshold event must be specified in PMLCA*n*[EVENT]. For this example, the duration threshold value is scaled by two because PMLCB*n*[TBMULT] is one. All other features are disabled by clearing the appropriate fields.

Any non-threshold event can use the burstiness feature. For burstiness counting, values for PMLCA*n*[BSIZE,BGRAN,BDIST] and PMLCB*n*[TBMULT] must be specified.

Table 19-12. Register Settings for Counting Examples

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMGC0	FAC	0	0	0	0
	PMIE	1	1	1	1
	FCECE	1	1	1	1
PMLCA*n*	FC	0	0	0	0
	CE	1	1	1	1
	EVENT	89	68	39	2
	BSIZE	0	0	0	5
	BGRAN	0	0	0	1
	BDIST	0	0	0	8
PMLCB*n*	TRIGONSEL	0	3	0	0
	TRIGOFFSEL	0	5	0	0
	TRIGONCNTL	0	1	0	0
	TRIGOFFCNTL	0	2	0	0
	TBMULT	0	0	0	0
	THRESHOLD	0	0	3	0

Chapter 20
Debug Features and Watchpoint Facility

This chapter describes all customer-visible debug modes of the MPC8555E integrated device. The debug features on the MPC8555E pertain to these three interfaces: the PCI interface (only when in 64-bit PCI mode), the local bus controller (LBC), and the DDR SDRAM interface. In addition to the external interfaces, the MPC8555E provides triggering capabilities based on user-programmable events. The watchpoint and trace buffer also provide some visibility to internal buses. This chapter also describes context ID registers useful for software debug and describes the JTAG access port signals that comply with the IEEE 1149.1 boundary-scan specification.

20.1 Introduction

As shown in the block diagram of Figure 20-1, the MPC8555E device provides the following debug features (listed with references to sections of this chapter that describe them):

- PCI interface debug (Section 20.4.2, “PCI Interface Debug”)
- DDR SDRAM interface debug (Section 20.4.3, “DDR SDRAM Interface Debug”)
- Local bus controller (LBC) debug (Section 20.4.4, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 20.4.5, “Watchpoint Monitor,” and Section 20.4.6, “Trace Buffer”)

20.1.1 Overview

As shown in Figure 20-1, debug information is provided through the following interfaces: PCI, LBC, and DDR SDRAM. Limited visibility, through a 256 × 64 trace buffer, is also provided for the processor core interface. This visibility into internal device operation is useful for debugging application software through inverse assembly and reconstruction of the fetch stream.

The combination of a source ID (MSRCID[0:4]) and a data-valid signal (MDVAL) indicates that meaningful debug information is visible on either the local bus or DDR SDRAM interfaces. A logic analyzer can be programmed to capture data based on the values of MSRCID[0:4] and MDVAL.

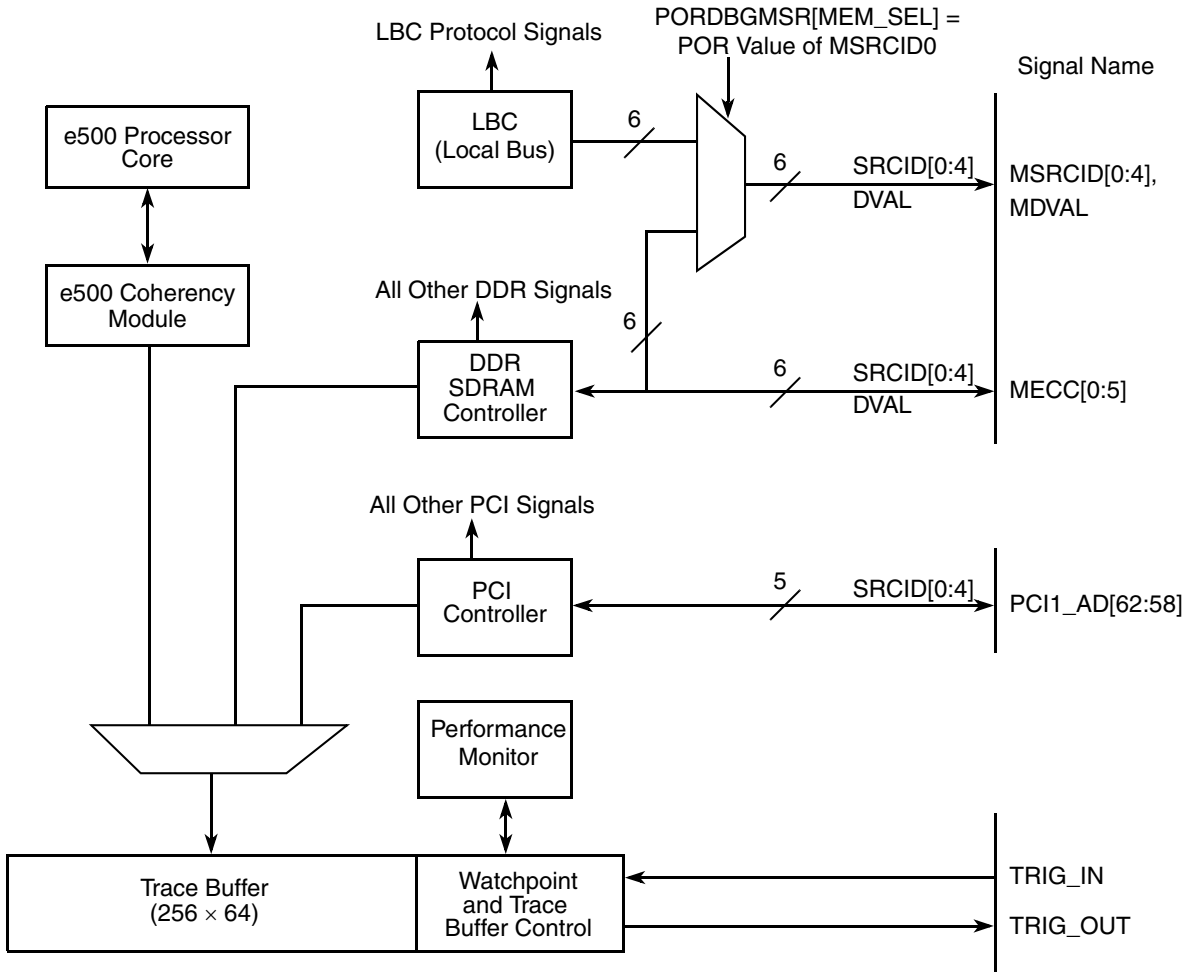


Figure 20-1. Debug and Watchpoint Monitor Block Diagram

Other system debugging is supported by the programmable triggering of the watchpoint monitor and trace buffer. Both can be triggered from one of the following three sources:

- Each other
- A performance monitor event
- An external source (through TRIG_IN)

The watchpoint monitor can be configured to assert TRIG_OUT when a programmed event occurs. The two context ID registers, described in Section 20.3.3, "Context ID Registers," are useful for software debug.

20.1.2 Features

The principal features of the debug modes and the watchpoint monitor are as follows:

- PCI1 interface: transaction source ID driven onto PCI1_AD[62:58] (only when in 64-bit PCI mode)

- LBC and DDR interface source ID and data-valid indicators
  - LBC or DDR SDRAM source ID can be selected to be driven onto MSRCID[0:4]
  - Source ID and data-valid indicators can be selected to be driven onto the error correcting code (ECC) pins of the DDR interface
- Watchpoint monitor that supports:
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocked with performance monitor to use its large number of counters
- Trace buffer features that supports:
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocked with performance monitor to use its large number of counters
  - 256-entry trace buffer, 64 bits each
  - Programmable trace start and stop
  - Can function as a second watchpoint monitor
- Context ID registers that can be programmed to trigger events

20.1.3 Modes of Operation

The PCI, LBC, and DDR SDRAM interfaces all have debug modes, which are controlled by values on configuration inputs during the power-on reset (POR) sequence, as shown in Table 20-1.The DDR controller can also drive debug information on either MSRCID[0:4] or MECC[0:5]. See Section 20.4.1, “Source and Target ID,” for additional information about the source ID information driven on the debug signals in these modes.

Note that both the watchpoint monitor and trace buffer also operate in a variety of modes.

Table 20-1. POR Configuration Settings and Debug Modes

Configuration Signal	POR Value	Effect	Reference
MSRCID0	0	Local bus SDRAM information appears on MSRCID[0:4] and MDVAL.	20.1.3.1/20-4
	1	Default value (internal pull-up resistor). DDR SDRAM information appears on MSRCID[0:4] and MDVAL.	
MSRCID1	0	MECC[0:4] operate in debug mode and provide memory debug source ID and MECC5 provides data-valid information.	20.1.3.2/20-4
	1	Default value (internal pull-up resistor). MECC[0:4] operate in normal mode and provide DDR SDRAM error correcting code information.	
$\overline{\text{PCI1\_GNT3}}$	0	The PCI interface operates in debug mode. The source ID information appears on the high-order address bits (PCI1_AD[62:58]) during the bus command phase.	20.1.3.3/20-4
	1	Default value (internal pull-up resistor). The PCI interface operates in normal mode.	

20.1.3.1 Local Bus (LBC) Debug Mode

The LBC and the DDR SDRAM controller can drive debug information (source ID and data-valid indicator) onto MSRCID[0:4] and MDVAL As shown in Table 20-1, the MSRCID0 value during POR controls multiplexing. If MSRCID0 is low when sampled during POR, the local bus SDRAM information appears on MSRCID[0:4] and MDVAL; otherwise, the DDR SDRAM debug information is presented.

20.1.3.2 DDR SDRAM Interface Debug Modes

MSRCID1 is sampled during POR to multiplex either ECC or debug information on the ECC pins of the DDR SDRAM interface. As shown in Table 20-1, if MSRCID1 is low during POR, the ECC pins operate in debug mode and provide memory debug source ID and data-valid information. MSRCID1 must be pulled low during POR to use the ECC pins in debug mode. If MSRCID1 is unconnected, an internal pull-up resistor ensures the ECC pins always source DDR SDRAM error correcting code information as their default power-on reset configuration.

NOTE

If the DDR ECC pins are in debug mode (configured for debug during POR), ECC checking is disabled in the memory controller. In this case, MECC[0:4] do not provide ECC information and must not be connected to SDRAM devices.

20.1.3.3 PCI Interface Debug Modes

If $\overline{\text{PCI1\_GNT3}}$ is low when sampled during POR, the PCI interface operates in debug mode. PCI debug mode is only possible when the PCI1 is configured for a 64-bit interface. In this mode, the source ID information appears on the high-order address bits (PCI1_AD[62:58]) during the bus command phase. See Section 20.4.2, “PCI Interface Debug,” for more information.

20.1.3.4 Watchpoint Monitor Modes

The watchpoint monitor supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The watchpoint monitor triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The watchpoint monitor waits for a specific event before enabling (arming) the trigger logic. The monitor does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Assert TRIG_OUT on hit—The debug block can be programmed to assert the TRIG_OUT signal when a programmed watchpoint monitor event occurs. This signal can be used to trigger a logic analyzer.

20.1.3.5 Trace Buffer Modes

The trace buffer supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The trace buffer triggers as soon as the first trigger event occurs.

- Wait for trigger arming (two-level triggering)—The trace buffer waits for a specific event before enabling (arming) the trigger logic. The trace buffer does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Specific interface selection—The trace buffer can be programmed to trace one of several internal interfaces.
- Specific event selection—The trace buffer can be programmed to trace on the occurrence of one or several concurrent events.
- Specific trace selection—To facilitate trace data filtering, the trace buffer can be configured to capture data under the following conditions:
  - On every cycle in which a valid transaction is present on the selected interface
  - Only when a watchpoint monitor event occurs
  - Only when the programmed trace event is detected
- Programmable trace stop—The trace buffer may be programmed to stop tracing when a programmed stop-tracing event occurs or when the 256-entry buffer is full.

20.2 External Signal Description

This section provides information about all the external signals associated with the various MPC8555E debug functions.

As shown in Table 20-1, the MPC8555E has several signals that are sampled during POR to determine the configuration of the phase-locked loop clock mode and the ROM, Flash, and dynamic memory. See Chapter 4, “Reset, Clocking, and Initialization.”

To facilitate system testing, the MPC8555E provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section also describes JTAG TAP signals.

20.2.1 Overview

All the signals associated with device debug features are summarized in Table 20-2, listed with a reference to the page number of the section with more information. The detailed descriptions are contained in Table 20-2. Some signals (the MECC bus, for example) are additionally described in other chapters, but are described here also for completeness, with emphasis on their debugging utility.

Table 20-2. Debug, Watchpoint, and Test Signal Summary

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
MDVAL	Memory data-valid	Debug	Selectable data-valid signal from either DDR SDRAM controller or LBC.	1	O	20-7
MECC[0:7]	DDR error correcting code	DDR SDRAM	In debug mode, the high-order 6 bits carry debug information (transaction source ID and data-valid indication).	0x08	O[1]	20-7
MSRCID[0:1]	Memory source ID	Debug	Selectable transaction source ID from either DDR SDRAM controller or local bus controller.	Reset_cfg	O	20-7
MSRCID[2:4]				111	O	20-7

Table 20-2. Debug, Watchpoint, and Test Signal Summary (continued)

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
TRIG_IN	Trigger in	Debug	Trigger for various function in the watchpoint monitor and trace buffer.	1	I	20-8
TRIG_OUT	Trigger out	Debug	Can be used externally for triggering a logic analyzer. Additionally, it can be used for observing system ready indication. Functions are multiplexed onto this signal depending on TOSR[SEL] (see Table 20-25).	1	O	20-8
PCI1_AD [62:58]	PCI address	Debug	In debug mode these pins carry the source ID of the transaction.	0000_0	O	20-7
TCK	Test clock	Debug	Clock for JTAG testing. Internally pulled up.	1	I	20-8
TDI	Test data input	Debug	Serial input for instructions and data to the JTAG test subsystem. Internally pulled up.	1	I	20-8
TDO	Test data output	Debug	Serial data output for the JTAG test subsystem. High impedance except when scanning out data.	Hi Z	O	20-8
TMS	Test mode select	Debug	Carries commands to the TAP controller for boundary scan operations. Internally pulled up.	1	I	20-8
$\overline{\text{TRST}}$	Test reset	Debug	Resets the TAP controller asynchronously.	—	I	20-8
THERM[0:1]	Thermal resistor access	Test	These pins tie directly to an internal resistor whose value varies linearly with temperature.	—	I	20-8
$\overline{\text{TEST\_SEL0}}$ (MPC8555E) TEST_SEL0 (MPC8541E)	Test select 0	Test	Factory test. Must be negated (pulled high on an MPC8555E; pulled low on an MPC8541E) for normal operation.	—	I	20-8
TEST_SEL1	Test select 1	Test	Factory test. Must be negated for normal operation.	—	I	20-8
$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory Test. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for proper treatment.		I	20-8
L1_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for proper treatment.		I	20-8
L2_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for proper treatment.		I	20-8

[1] While these signals are normally bidirectional, when sourcing debug information they are output only.

20.2.2 Detailed Signal Descriptions

This section describes the details of the debug, watchpoint monitor, and JTAG test signals.

20.2.2.1 Debug Signals—Details

Table 20-3 describes all signals associated with device debug modes.

Table 20-3. Debug Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDVAL	O	Memory data-valid. Indicates when valid data is available. May be used by a logic analyzer to capture the data on the data bus.	
		State Meaning	Asserted—Indicates that data is valid on the data bus during the current clock cycle. When the DDR SDRAM interface is selected to source information on MDVAL, this signal is valid for every cycle that data is driven or received on the DDR SDRAM interface. When the LBC is selected, this signal is valid for every cycle that data is driven or received on the local bus interface. The assertion of this signal may be used by a logic analyzer to capture data.
		Timing	Asserted/Negated—Referenced to the selected interface, (DDR or local bus). Asserts when data is valid. Assertions are held for the duration of the transfer. Read data timing is similar to MA. Write data timing is similar to the output MDQ.
MECC[0:7]	O	Memory ECC. DDR error checking and correcting. The normally bidirectional operation of the memory ECC (MECC) bus is described in Section 9.5.12, "Error Checking and Correcting (ECC)." This bus is used for debug functions when MSRCID1 is sampled low during POR. In debug mode, the high-order 5 bits (MECC[0:4]) may be used to provide the transaction source ID and MECC5 can be used as the data-valid indicator. In debug mode, MECC[0:5] is constantly driven with debug information and must be disconnected from the DDR memory's ECC pins.	
		State Meaning	Asserted/Negated—In debug mode, MECC[0:5] is always driven. The source ID values appear during $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ cycles. A value of 0x1F (all ones) is driven during cycles other than $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$. The data-valid indicator appears when data is being received or driven on the pins.
		Timing	Driven every cycle in debug mode.
MSRCID[0:4]	O	Memory source ID. Attribute signals associated with the memory interface that indicate the source ID for a transaction on an SDRAM interface. The SDRAM interface, DDR or local bus, to which the debug information applies is specified during POR with MSRCID0 as shown in Table 20-1. Two of these signals serve as reset configuration input signals.	
		State Meaning	Asserted/Negated—In debug mode, always driven with the value of the source ID. The source ID has a value of 0x1F for cycles other than $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$. The encodings shown in Table 20-26 provide detailed information about a memory transaction.
		Timing	Driven every cycle in debug mode. Similar timing to MA.
PCI1_AD[62:58]	O	PCI Address. Provides transaction source ID for the current PCI bus transaction.	
		State Meaning	Asserted/Negation—In debug mode, always driven with the value of the transaction source ID.
		Timing	Driven every cycle in debug mode.

20.2.2.2 Watchpoint Monitor Trigger Signals—Details

Table 20-4 shows detailed descriptions of the watchpoint monitor and trace buffer signals.

Table 20-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TRIG_IN	I	Trigger in. Can be used to trigger the watchpoint and trace buffers. Note this is an active-high (rising-edge triggered) signal.	
		State Meaning	Asserted—Indicates that a programmed/armed external event has been detected. Assertion may be used internally to trigger trace buffers and watchpoint mechanisms.
		Timing	Assertion/Negation—The MPC8555E interprets TRIG_IN as asserted on detection of the rising edge. It may occur at any time. Must remain asserted for at least 3 system clocks to be recognized internally.
TRIG_OUT	O	Trigger out. Function determined by TOSR[SEL]. When TOSR[SEL] is non-zero, it can be used for triggering external devices, like a logic analyzer, with either the watchpoint monitor, the trace buffer, or the performance monitor as trigger sources. When TOSR[SEL] is cleared, TRIG_OUT is multiplexed with READY, which indicates the operational readiness of the device (running or in low-power or debug modes). See Chapter 4, “Reset, Clocking, and Initialization,” and Chapter 18, “Global Utilities,” for more details about reset, low-power, and debug states.	
		State Meaning	Asserted—When TOSR[SEL] is all zeros, serves as the READY signal, indicating that the device is not in a low-power or debug mode and that it has emerged from reset. SEL ≠ 0 indicates that a programmed trigger event has occurred. Negation—No final watchpoint match condition
		Timing	Assertion may occur at any time. Remains asserted for at least 3 system clocks

20.2.2.3 Test Signals—Details

Table 20-5 shows detailed descriptions of the JTAG test signals.

Table 20-5. JTAG Test and Other Signals—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. 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 standard 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 standard for more details.

Table 20-5. JTAG Test and Other Signals—Detailed Signal Descriptions

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 standard 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 standard 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 MPC8555E. 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 standard for more details.
$\overline{\text{LSSD\_MODE}}$	I	Used for factory test. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for proper treatment.	
L1_TSTCLK	I	Used for factory test. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for proper treatment.	
L2_TSTCLK	I	Used for factory test. Refer to the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for proper treatment.	
THERM[0:1]	I	These signals provide access to an internal resistor that has a value that varies linearly with temperature. The actual value for the resistor varies from device to device, but the linear relationship between temperature and resistance is consistent. See the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications* for more information on how to accurately measure the junction temperature of a device. Note that this thermal resistor is intended for engineering development only.	
$\overline{\text{TEST\_SEL0}}$ (MPC8555E) TEST_SEL0 (MPC8541E)	I	Used for factory test. This test signal is active-low on the MPC8555E device but active-high on the MPC8541E device. For normal operation, it should be negated (pulled high on an MPC8555E; pulled low on an MPC8541E).	
TEST_SEL1	I	Used for factory test. Should be negated for normal operation	

20.3 Memory Map/Register Definition

Table 20-6 shows the memory-mapped debug and watchpoint registers of the MPC8555E. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 20-6. Debug and Watchpoint Monitor Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	20.3.1.1/20-10
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	20.3.1.1/20-10
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	20.3.1.2/20-12
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	20.3.1.3/20-13
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	20.3.1.4/20-13
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	20.3.1.5/20-15
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	20.3.2.1/20-15
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	20.3.2.1/20-15
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	20.3.2.2/20-18
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	20.3.2.3/20-18
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	20.3.2.4/20-18
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	20.3.2.5/20-19
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	20.3.2.6/20-20
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	20.3.2.7/20-21
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	20.3.2.8/20-21
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	20.3.3.1/20-22
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	20.3.3.2/20-22
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	20.3.4.1/20-23

20.3.1 Watchpoint Monitor Register Descriptions

The following sections describe the control registers for the watchpoint monitor facility.

20.3.1.1 Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)

The watchpoint monitor control registers (WMCR0, WMCR1), shown in Figure 20-2 and Figure 20-3, control the specification of watchpoint monitor events.

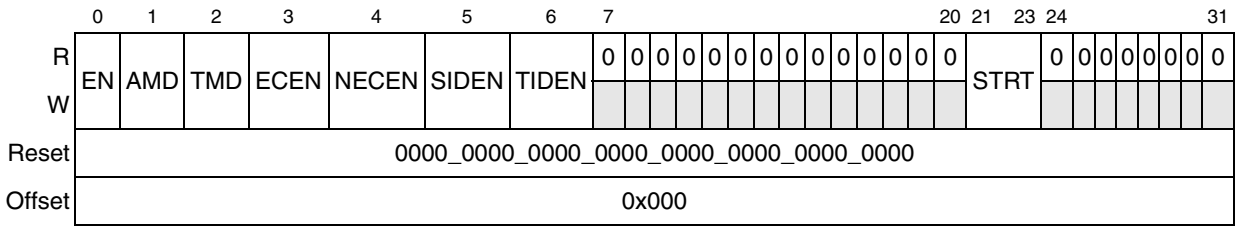


Figure 20-2. Watchpoint Monitor Control Register 0 (WMCR0)

Table 20-7 describes WMCR0 fields.

Table 20-7. WMCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 Watchpoint monitor events are not flagged. 1 A watchpoint monitor event is flagged.
1	AMD	Address match disable. Qualifies address match as a watchpoint event criterion. 0 Address matching is used to recognize a watchpoint event. 1 Address matching does not affect watchpoint event detection.
2	TMD	Transaction match disable. Qualifies transaction type match (as defined in WMCR1[IFSEL] and WMTMR) as a watchpoint event criterion. 0 A transaction type match is used to recognize watchpoint events. 1 A transaction type match does not affect watchpoint event detection.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 20.3.3, “Context ID Registers.” 0 Current context match does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 20.3.3, “Context ID Registers.” 0 The failure of a current context match does not affect watchpoint event detection 1 Watchpoint events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Source ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(SID) value.
6	TIDEN	Target ID enable 0 Target ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(TID) value.
7–20	—	Reserved

Table 20-7. WMCR0 Field Descriptions (continued)

Bits	Name	Description
21–23	STRT	Start condition. Specifies the event that arms the watchpoint monitor to start looking for the programmed event. 000 No event. Armed immediately 001 Trace buffer event is detected 010 Performance monitor signals overflow 011 TRIG_IN transitions from 0 to 1 100 TRIG_IN transitions from 1 to 0 101 Current context ID equals programmed context ID 110 Current context ID is not equal to programmed context ID 111 Reserved
24–31	—	Reserved

Figure 20-3 shows the WMCR1.

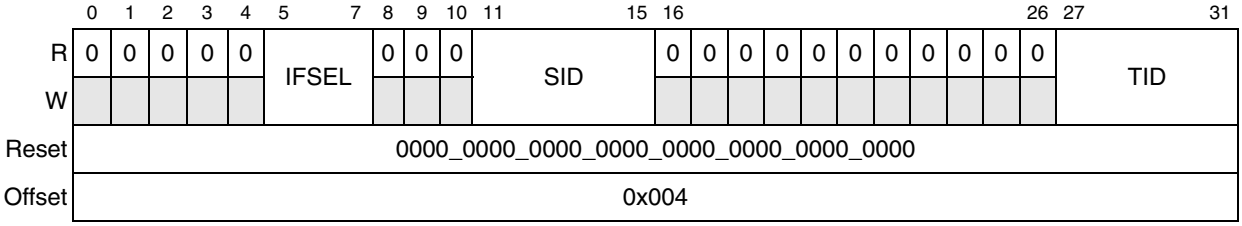


Figure 20-3. Watchpoint Monitor Control Register 1 (WMCR1)

Table 20-8 describes the WMCR1 fields.

Table 20-8. WMCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Selects the address, transaction type (as defined in WMTMR), and other attributes to be used for comparison 000 Select e500 coherency module (ECM) dispatch interface 001 Select internal DDR SDRAM interface 010 Select internal PCI outbound interface 011–111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with WMCR0[SIDEN]. For a definition of the source ID, see Table 20-26.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with WMCR0[TIDEN]. For a definition of the target ID see Table 20-26.

20.3.1.2 Watchpoint Monitor Address Register (WMAR)

The watchpoint monitor address register (WMAR), shown in Figure 20-4, contains the address to match against if WMCR[AMD] is clear. Note that the transaction address is qualified with the bits described in

Section 20.3.1.3, "Watchpoint Monitor Address Mask Register (WMAMR)," before being compared with WMAR. Note also that the contents of WMAR are not qualified with WMAMR.

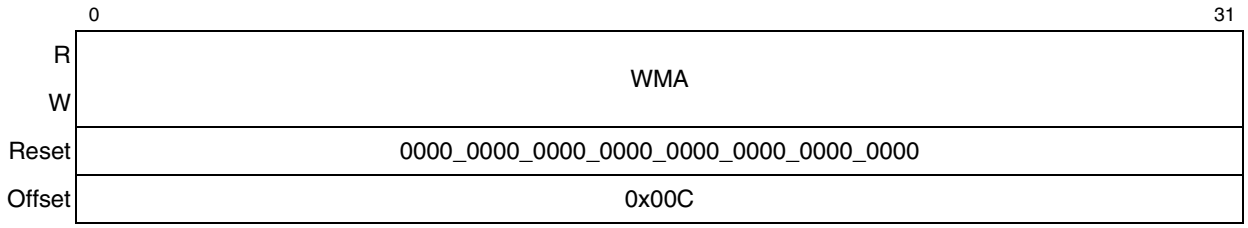


Figure 20-4. Watchpoint Monitor Address Register (WMAR)

Table 20-9 describes the WMAR fields.

Table 20-9. WMAR Field Descriptions

Bits	Name	Description
0–31	WMA	Watchpoint monitor address.

20.3.1.3 Watchpoint Monitor Address Mask Register (WMAMR)

The watchpoint monitor address mask register (WMAMR) shown in Figure 20-5 contains the mask that is applied to a transaction address before the address is compared with WMAR.

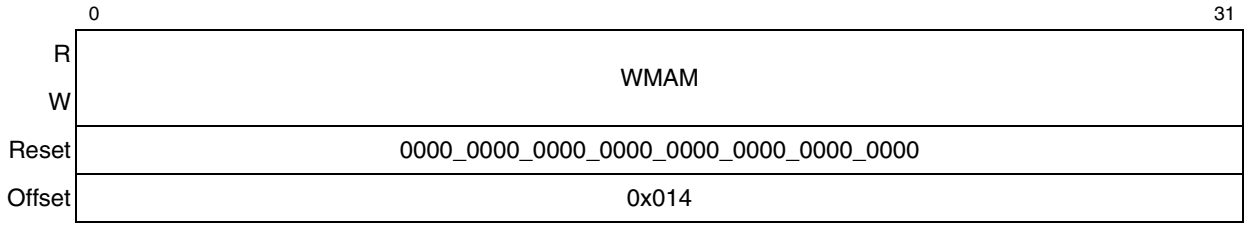


Figure 20-5. Watchpoint Monitor Address Mask Register (WMAMR)

Table 20-10 describes the WMAMR fields.

Table 20-10. WMAMR Field Descriptions

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. A value of zero masks the address comparison for the corresponding address bit.

20.3.1.4 Watchpoint Monitor Transaction Mask Register (WMTMR)

The watchpoint monitor transaction mask register (WMTMR), shown in Figure 20-6, specifies which transaction types to monitor. WMTMR allows users to qualify watchpoint events specifically with any combination of transaction types. As shown in Table 20-12, each bit represents as many as four separate transaction types; one for each interface. Setting a bit enables watchpoint monitoring for the corresponding transaction types.

Because the supported transaction types vary by interface, the type designated by a WMTMR field also depends on the interface specified by WMCR1[IFSEL]. Table 20-12 lists transaction types associated with each WMTMR bit by interface.

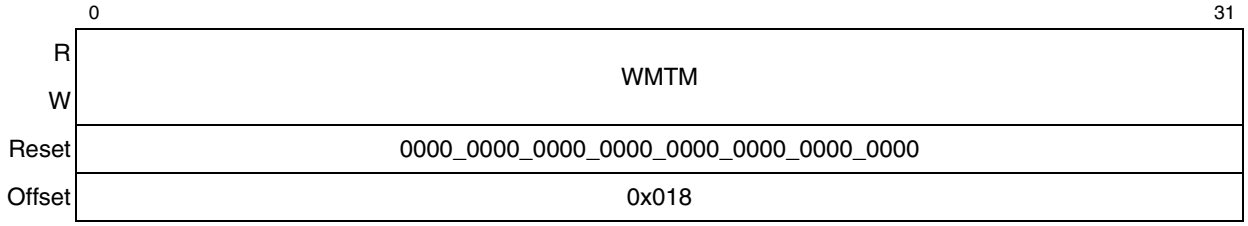


Figure 20-6. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 20-11 describes the WMTMR fields.

Table 20-11. WMTMR Field Descriptions

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit corresponds to a transaction type as defined in Table 20-12. The transaction associated with any particular bit may be different depending on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when WMCR0[TMD] = 0.

The following table, Table 20-12, defines the transactions associated with each transaction mask bit for the different interfaces supported by the watchpoint monitor.

Table 20-12. Transaction Types By Interface

Bits	e500 Coherency Module Dispatch	DDR Controller	PCI Outbound Request
0	Write with local processor snoop	Write	Memory write
1	Write with no local processor snoop	—	I/O write
2	Write with allocate(L2 stashing)	Write with allocate	—
3	Write with allocate and lock (L2 stashing with locking)	Write with allocate and lock	—
4–7	Reserved		
8	Read with local processor snoop	Read	Memory read
9	Read with no local processor snoop	—	I/O read
10	Read with unlock	Read with unlock	—
11–15	Reserved		
16	Atomic clear	ATOMIC clear	—
17	Atomic set	ATOMIC set	—
18	Atomic decrement	ATOMIC decrement	—
19	Atomic increment	ATOMIC increment	—
20–31	Reserved		

20.3.1.5 Watchpoint Monitor Status Register (WMSR)

The watchpoint monitor status register (WMSR), shown in Figure 20-7, indicates the state of the watchpoint monitor.

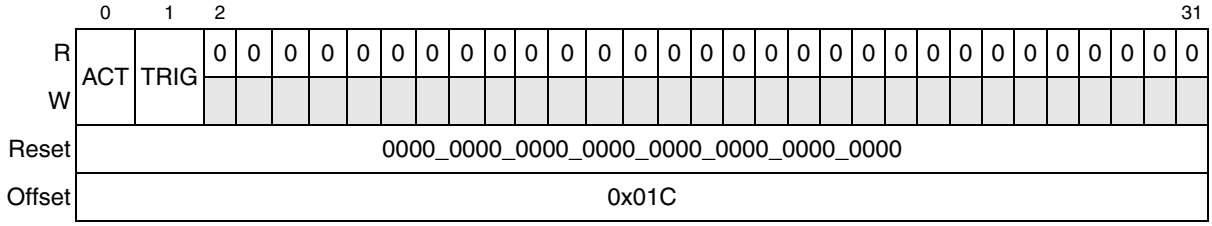


Figure 20-7. Watchpoint Monitor Status Register (WMSR)

Table 20-13 describes the WMSR fields.

Table 20-13. WMSR Field Descriptions

Bits	Name	Description
0	ACT	Active 0 The start triggering event has not occurred; watchpoint monitor is not armed. 1 The start triggering event has occurred; watchpoint monitor is armed.
1	TRIG	Triggered 0 The programmed event in WMCR0 has not yet been triggered. 1 The programmed event in WMCR0 has been triggered at least once.
2–31	—	Reserved

20.3.2 Trace Buffer Register Descriptions

The following sections describes the trace buffer registers.

20.3.2.1 Trace Buffer Control Registers (TBCR0, TBCR1)

The trace buffer control registers (TBCR0, TBCR1), shown in Figure 20-8 and Figure 20-9, specify trace buffer events.

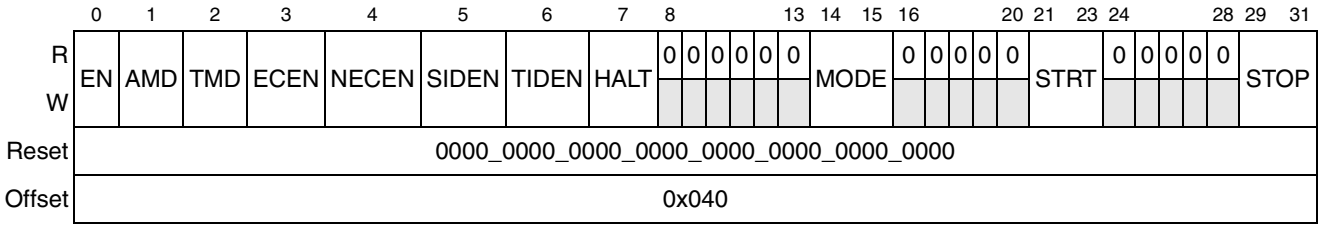


Figure 20-8. Trace Buffer Control Register 0 (TBCR0)

Table 20-14 describes the TBCR0 fields.

Table 20-14. TBCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 The trace buffer facility is disabled. 1 The trace buffer facility is enabled.
1	AMD	Address match disable 0 The address match is used to qualify a trace buffer event. 1 The address match is ignored when detecting a trace buffer event.
2	TMD	Transaction match disable 0 The transaction type match is used to qualify a trace buffer event. 1 The transaction type match is ignored when detecting a trace buffer event.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 20.3.3, “Context ID Registers.” 0 Current context match does not affect trace buffer event detection 1 Trace buffer events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 20.3.3, “Context ID Registers.” 0 The failure of a current context match does not affect trace buffer event detection 1 trace buffer events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Trace buffer events ignore the programmed source ID value. 1 Trace buffer events are qualified by comparison with the programmed SID event value.
6	TIDEN	Target ID enable 0 Trace buffer events ignore the programmed TID event value. 1 Trace buffer events are qualified by comparison with the programmed TID event value. This comparison only applies when the ECM is selected for tracing (TBCR1[IFSEL] is all zeros).
7	HALT	Halt causes the trace buffer to stop tracing immediately.
8–13	—	Reserved
14–15	MODE	Trace mode. Specifies one of two trace modes. 00 Trace every valid transaction 01 Reserved 10 Trace only cycles in which a trace event is detected. Note that if EN and other TBCR0 fields are not properly programmed to specify a traceable event, tracing occurs for every valid address. 11 Reserved
16–20	—	Reserved

Table 20-14. TBCR0 Field Descriptions (continued)

Bits	Name	Description
21–23	STRT	Start condition. Specifies the event that arms the trace buffer to start looking for the programmed event 000 No event. Armed immediately 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID
24–28	—	Reserved
29–31	STOP	Trace stop mode. Specifies the event that stops the updating of the trace buffer after it has been started. Trace buffer only stops after it has been triggered at least once. 000 Buffer is full 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID

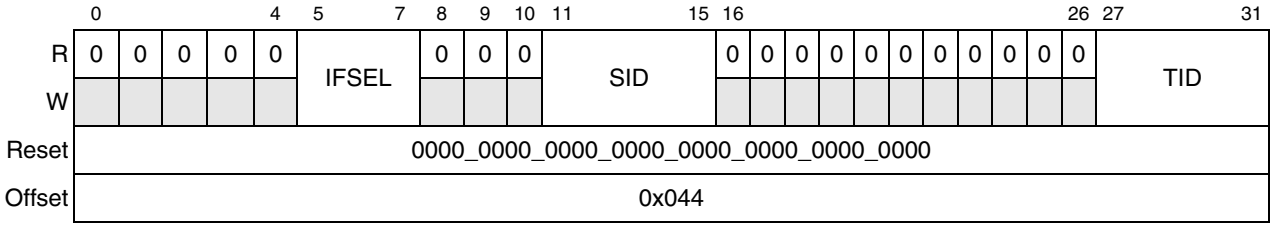


Figure 20-9. Trace Buffer Control Register 1 (TBCR1)

Table 20-15 describes the TBCR1 fields.

Table 20-15. TBCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Specifies the interface that sources information for both comparison/buffer control and buffer data capture. 000 Selects e500 coherency module (ECM) dispatch interface 001 Selects internal DDR SDRAM interface 010 Selects internal PCI outbound interface 011–111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with TBCR0[SIDEN]. The source ID is defined in Table 20-26.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with TBCR0[TIDEN]. The target ID is defined in Table 20-26.

20.3.2.2 Trace Buffer Address Register (TBAR)

The trace buffer address register (TBAR), shown in Figure 20-10, contains the address to match against (if TBCR0[AMD] is zero). The transaction address is qualified with the bits described in Section 20.3.2.3, "Trace Buffer Address Mask Register (TBAMR)," before being compared with TBAR. Note that the contents of TBAR are not qualified with TBAMR.

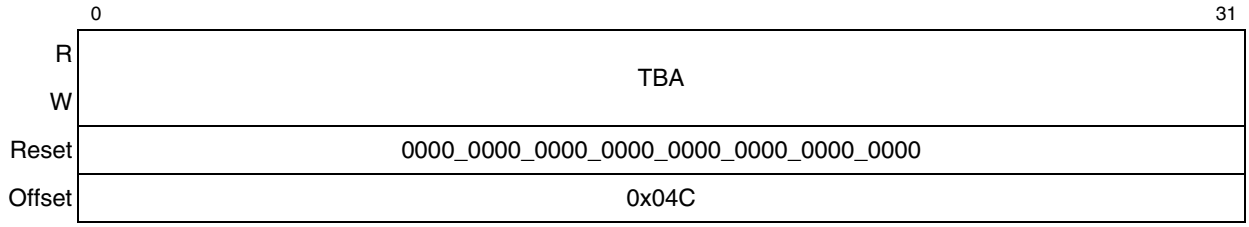


Figure 20-10. Trace Buffer Address Register (TBAR)

Table 20-16 describes the TBAR field.

Table 20-16. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address.

20.3.2.3 Trace Buffer Address Mask Register (TBAMR)

The trace buffer address mask register (TBAMR), shown in Figure 20-11, contains the mask that is applied to a transaction address before the address is compared with TBAR.

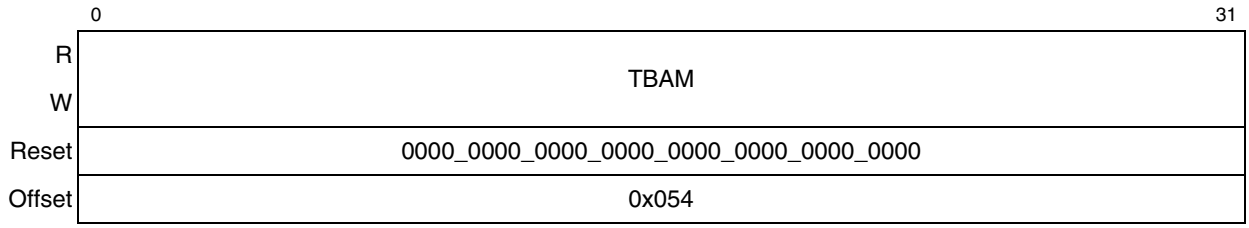


Figure 20-11. Trace Buffer Address Mask Register (TBAMR)

Table 20-17 describes the TBAMR field.

Table 20-17. TBAMR Field Descriptions

Bits	Name	Description
0–31	TBAM	Trace buffer address mask.A value of zero masks the address comparison for the corresponding address bit.

20.3.2.4 Trace Buffer Transaction Mask Register (TBTMR)

The trace buffer transaction mask register (TBTMR), shown in Figure 20-12, specifies which transaction types to monitor. Each bit in the TBTMR represents a transaction type on the selected interface. The transaction associated with any particular bit depends on the interface being monitored as specified by TBCR1[IFSEL]. Note that the transactions used for defining trace buffer events are the same as those

defined for watchpoint monitor events. Thus, Table 20-12 defines the transaction types associated with each interface. Setting a bit enables a hit when this transaction is matched (provided all other match criteria are met and TBCR0[TMD] is clear).

Different interfaces support different transaction types, and the same bit may represent different transaction types depending on the interface.

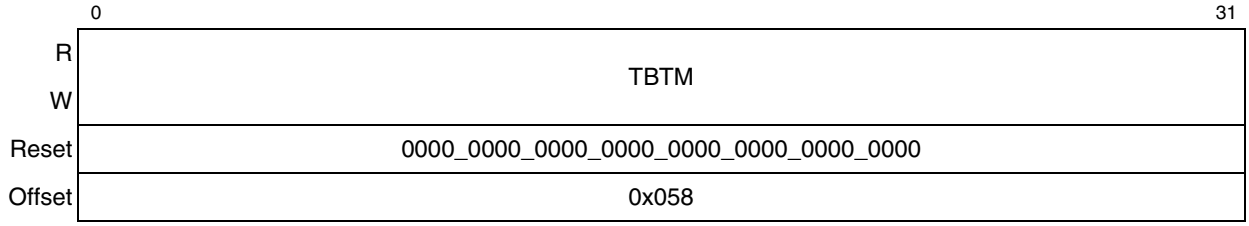


Figure 20-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 20-18 describes the TBTMR field.

Table 20-18. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTM	Trace buffer transaction mask. Each bit corresponds to a transaction type as defined in Table 20-12. The transaction associated with a bit depends on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when TBCR0[TMD] = 0.

20.3.2.5 Trace Buffer Status Register (TBSR)

The trace buffer status register (TBSR), shown in Figure 20-13, indicates the operational state of the trace buffer.

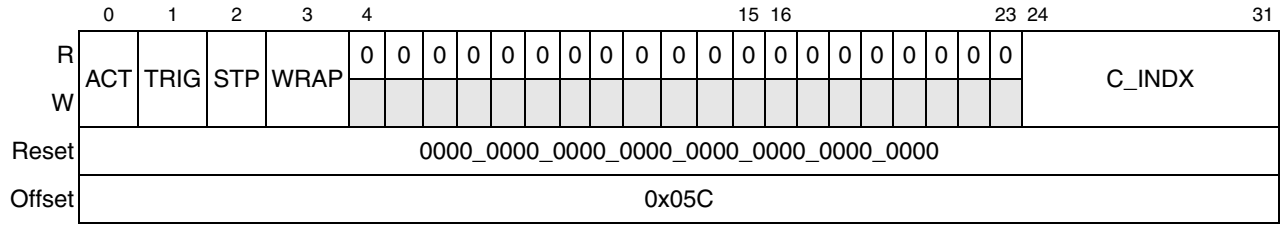


Figure 20-13. Trace Buffer Status Register (TBSR)

Table 20-19 describes the TBSR fields.

Table 20-19. TBSR Field Descriptions

Bits	Name	Description
0	ACT	Active. Indicates trace buffer activity 0 The start triggering event has not yet occurred. Trace buffer is not armed 1 The start triggering event has occurred. Trace buffer is armed
1	TRIG	Triggered. Indicates whether or not a programmed event has been triggered 0 The programmed event in TBCR0 has not yet been triggered. 1 The programmed event in TBCR0 has been triggered at least once.

Table 20-19. TBSR Field Descriptions (continued)

Bits	Name	Description
2	STP	Stopped. Indicates whether or not a trace buffer stop condition has been detected 0 No stop condition yet detected 1 The trace buffer has detected a stop condition and is no longer capturing events.
3	WRAP	Wrapped. Indicates that the trace buffer write pointer has wrapped to the beginning of the buffer at least once. Set when the last entry of the trace buffer is written 0 Pointer has not yet wrapped 1 Pointer has wrapped to the beginning at least once
4–23	—	Reserved
24–31	C_INDX	Current index. Represents the current value of the write pointer at the time TBSR was read. This value may be written by software to initialize the write pointer; however, software is not allowed to write the write pointer while the trace buffer is active. Writes are ignored while the trace buffer is active. It is recommended to write the status register before enabling the trace buffer in order to zero out any bits that might have been set during a prior run and to initialize the write pointer to zero.

20.3.2.6 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR) enables software to read or write the trace buffer. Each entry is 64 bits; therefore, it takes one write of TBACR and two reads of the access data register (TBADR and TBADHR) to read one 256-entry array entry. Similarly, it takes one write of TBACR and two writes of TBADR and TBADHR to write one array entry. Software can access any entry by writing the appropriate index into TBACR[INDX]. To read or write the buffer sequentially, starting with entry 0, the index must start with a value of 0 and increment every time a new entry is accessed.

TBACR is shown in Figure 20-14.

	0	1	2																			23	24	31
R	RD	WR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INDX	
W																								
Reset	0000_0000_0000_0000_0000_0000_0000_0000																							
Offset	0x060																							

Figure 20-14. Trace Buffer Access Control Register (TBACR)

Table 20-20 describes the TBACR fields.

Table 20-20. TBACR Field Descriptions

Bits	Name	Description
0	RD	Read command. When set, a trace buffer read is performed using the value of TBACR[INDX]. This bit is automatically cleared when the read is performed.
1	WR	Write command. When set, a trace buffer write is performed using the value of TBACR[INDX]. This bit is automatically cleared when the write is performed. A write occurs only if the trace buffer is not active: write requests are ignored while the buffer is active.
2–23	—	Reserved
24–31	INDX	Buffer index to read from or write into (0–255). Used in conjunction with TBACR[RD] and TBACR[WR].

20.3.2.7 Trace Buffer Access Data High Register (TBADHR)

The trace buffer access data high register (TBADHR), shown in Figure 20-15, contains the high-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]), or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

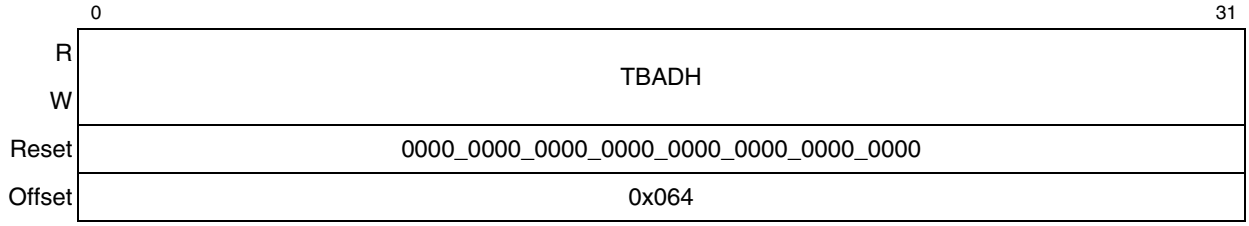


Figure 20-15. Trace Buffer Read High Register (TBADHR)

Table 20-21 describes TBADHR.

Table 20-21. TBADHR Field Descriptions

Bits	Name	Description
0–31	TBADH	Trace buffer access data high. The higher 32 bits of the data read from or to be written into the trace buffer, depending on whether the array is accessed with a read or a write.

20.3.2.8 Trace Buffer Access Data Register (TBADR)

The trace buffer access data register (TBADR), shown in Figure 20-16, contains the low-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]) or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

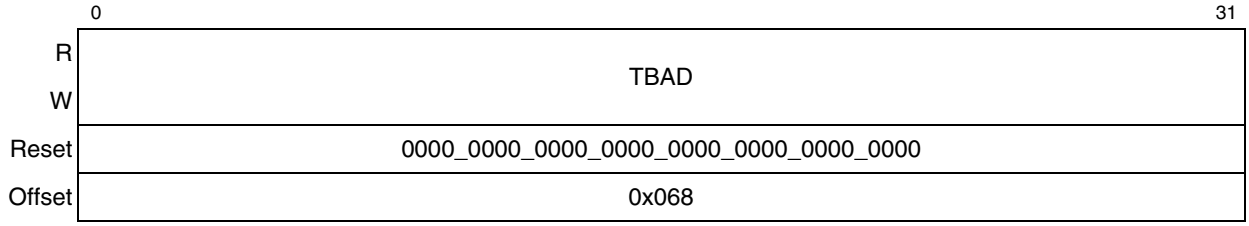


Figure 20-16. Trace Buffer Access Data Register (TBADR)

Table 20-22 describes the TBADR field.

Table 20-22. TBADR Field Descriptions

Bits	Name	Description
0–31	TBAD	Trace buffer access data. Corresponds to the lower 32 bits of the data read from the trace buffer or to be written into the trace buffer, depending on whether software is accessing the array with a read or a write.

20.3.3 Context ID Registers

This section describes the context ID registers. The current context ID register (CCIDR) and programmed context ID registers (PCIDR) are set by software and facilitate debugging complex software.

20.3.3.1 Programmed Context ID Register (PCIDR)

The programmed context ID register (PCIDR), shown in Figure 20-17, contains the user-programmed context ID. This register can be configured to trigger watchpoint events when its value matches the current context ID register (CCIDR), as controlled by WMCR0[ECEN] and WMCR0[NECEN]. See Section 20.3.1.1, "Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)," for more information.

	0	31
R W	PCID	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x0A0	

Figure 20-17. Programmed Context ID Register (PCIDR)

Table 20-23 describes the PCIDR field.

Table 20-23. PCIDR Field Descriptions

Bits	Name	Description
0–31	PCID	Programmed context ID. Contains the user-programmed context ID. Compared with current context ID for context-sensitive event triggering

20.3.3.2 Current Context ID Register (CCIDR)

The current context ID register (CCIDR), shown in Figure 20-18, contains the current context ID. This register is written by software after a context switch and can be used to trigger events when compared with the programmed context ID register (PCIDR).

	0	31
R W	CCID	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x0A4	

Figure 20-18. Current Context ID Register (CCIDR)

Table 20-24 describes the CCIDR field.

Table 20-24. CCIDR Field Descriptions

Bits	Name	Description
0–31	CCID	Current context ID. Set by user software. Typically loaded immediately following a context switch. Compared with user-programmed context ID for context-sensitive event triggering

20.3.4 Trigger Out Function

TRIG_OUT provides a convenient mechanism for triggering external system monitors and diagnostic equipment such as logic analyzers. Note that READY is multiplexed with TRIG_OUT. See the last paragraph of Section 4.4.2, "Power-On Reset Sequence," for more information about READY functionality.

When the trace buffer hit is selected by TOSR[SEL], TRIG_OUT is only meaningful if the trace buffer control register 0 (TBCR0) is properly configured to hit on a traceable event. The same holds true for the watchpoint monitor when the watchpoint monitor is selected by TOSR[SEL].

20.3.4.1 Trigger Out Source Register (TOSR)

The trigger out source register (TOSR) shown in Figure 20-19 specifies the source for TRIG_OUT. The three event-trigger sources are the following:

- The watchpoint monitor
- The trace buffer
- The performance monitor

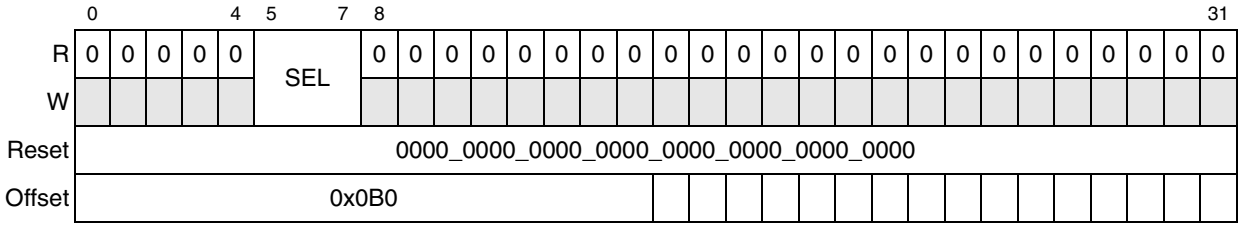


Figure 20-19. Trigger Out Source Register (TOSR)

Table 20-25 describes the TOSR fields.

Table 20-25. TOSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	SEL	Select. Selects the source for TRIG_OUT 000 READY signal. Multiplexed with TRIG_OUT. Basic device state indicator. READY asserts whenever the device is not in reset or not asleep. See Chapter 4, "Reset, Clocking, and Initialization," for more details about the reset sequence, and Chapter 18, "Global Utilities," for more information about power management states. 001 Selects the watchpoint monitor hit indication 010 Selects the trace buffer hit indication 011 Selects the performance monitor overflow indication
8–31	—	Reserved

20.4 Functional Description

The debug features on the MPC8555E use the PCI interface, the LBC interfaces, and the DDR SDRAM interface.

20.4.1 Source and Target ID

Debug information that is common to all the interfaces is the source ID (SID). The transaction source ID provides enough information to determine which block or port originated a transaction including the distinction between instruction and data fetches from the processor core. Table 20-26 shows the values and interpretation for the 5-bit SID field. Note that the table also includes ports that are only slaves, such as local memory. These ports are always targets. As such, the value shown represents a target ID (TID) and not a source ID. For ports that can function in both capacities, the value indicates source ID when mastering transactions, and target ID when responding as slave. The TID field is only meaningful when one of the following participates in the transaction:

- The e500 coherency module (ECM) dispatch bus
- The watchpoint monitor (WMCR1[IFSEL] = 000)
- The trace buffer (TBCR1[IFSEL] = 000)

Table 20-26. Source and Target ID Values

Value (Hex)	Source (or Target) Port	Value (Hex)	Source (or Target) Port
00	PCI1	10	Local processor (instruction fetch)
01	PCI2	11	Local processor (data fetch)
02	Reserved	12	Reserved
03	Reserved	13	Reserved
04	Local bus controller	14	CPM
05	Reserved	15	DMA

Table 20-26. Source and Target ID Values (continued)

Value (Hex)	Source (or Target) Port	Value (Hex)	Source (or Target) Port
06	Reserved	16	Reserved
07	Security	17	System access port (SAP)
08	Configuration space	18	TSEC1
09	Reserved	19	TSEC2
0A	Boot sequencer	1A	Reserved
0B	Reserved	1B	Reserved
0C	Reserved	1C	Reserved
0D	Reserved	1D	Reserved
0E	Reserved	1E	Reserved
0F	Local space (DDR)	1F	Non valid port indicator (reserved for debug info)

20.4.2 PCI Interface Debug

If $\overline{\text{PCI1\_GNT3}}$ is low when sampled during POR, the PCI interface operates in debug mode. PCI debug mode is only possible when PCI1 is configured as a 64-bit interface. In debug mode the source ID appears on the high-order address bits (PCI1_AD[62:58]) during the bus command phase of a PCI transaction. The bus command phase occurs either during the first cycle that $\overline{\text{PCI1\_FRAME}}$ is asserted, or, in the case of addresses greater than 32 bits, after a dual-address cycle phase. In either case, the debug information appears on the highest order address bits while $\overline{\text{PCI1\_FRAME}}$ is asserted and both $\overline{\text{PCI1\_IRDY}}$ and $\overline{\text{PCI1\_TRDY}}$ are negated.

When accessing the low 4 Gbytes of PCI address space for which no dual-address cycle is needed, the debug information appears during the first (and only) address phase on PCI1_AD[62:58]. Whenever a dual-address cycle must be run, (addresses above 4 Gbytes) the debug information appears on PCI1_AD[62:58] during the second address cycle. In either case a logic analyzer should be configured to sample information on the first cycle of the assertion of $\overline{\text{PCI1\_FRAME}}$ and the cycle following a dual-address cycle command.

NOTE

Because they share the same pins, an entire 64-bit address and the debug information cannot be captured in a single cycle.

20.4.3 DDR SDRAM Interface Debug

The DDR interface has two debug modes distinguished by which pins drive the debug information. In one mode, debug information (source ID, data valid) is multiplexed onto the ECC pins; the other mode uses the debug pins.

20.4.3.1 Debug Information on Debug Pins

If MSRCID0 is high when sampled during POR, the debug information from the DDR SDRAM interface is driven on MSRCID[0:4] and MDVAL. This POR value is captured in PORDBGMSR[MEM_SEL] as described in Section 18.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)." In this mode, the source ID appears on MSRCID[0:4] during a $\overline{\text{RAS}}$ or $\overline{\text{CAS}}$ cycle. During any other cycle, the value of MSRCID[0:4] is all ones, which indicates idle cycles on the address/command interface. Similarly, MDVAL is asserted during valid data cycles on the DDR interface.

20.4.3.2 Debug Information on ECC Pins

If MSRCID1 is low when sampled during POR, debug information from the DDR SDRAM interface is selected to appear on MECC[0:5] as shown in Figure 20-1. In this mode, the ID value of the source port, (the source ID), appears on MECC[0:4] during a $\overline{\text{RAS}}$ or $\overline{\text{CAS}}$ cycle. During any other cycle the value of MECC[0:4] is all ones. A data-valid signal (DVAL) is driven on MECC5 during valid DDR SDRAM data cycles.

NOTE

In this mode, MECC[0:5] must be disconnected from all SDRAM devices to prevent contention on those lines.

20.4.4 Local Bus Interface Debug

If MSRCID0 is low when sampled during POR, the LBC is selected as the source for the debug information appearing on MSRCID[0:4] and MDVAL. For more information on this mode, see Section 13.1.3.2, "Source ID Debug Mode."

20.4.5 Watchpoint Monitor

The watchpoint monitor (WM) can be programmed to arm and trigger on many different events including any of the following:

- External event (through TRIG_IN)
- A trace buffer event
- A performance monitor overflow event
- A comparison of the current and programmed context ID registers.

A watchpoint event can be used in the following ways:

- Trigger a logic analyzer (using TRIG_OUT)
- Arm or trigger the trace buffer
- Trigger a performance monitor event.

The large counters available in the performance monitor block and the interlock between it and the watchpoint monitor support sophisticated debug scenarios.

A WM trigger event may be composed of several events programmed in the watchpoint monitor control registers (WMCR0–WMCR1). Because the watchpoint monitor is disabled by default during POR, these

registers must be initialized to make use of this debug feature. Note that the WM address mask register (WMAMR) and the type mask register (WMTMR) are cleared during POR. This means that the watchpoint monitor's default behavior following a power-on reset is to trigger on any address and no transaction type. The reset value of WMCR0[TMD] is 0 which means transaction matching is enabled but since no transaction is selected (WMTMR = 0), a match will never occur. Either the transaction matching must be disabled by setting WMCR0[TMD] to a value of 1, or valid transactions must be selected by setting one or more of the WMTMR bits to a value of 1.

20.4.5.1 Watchpoint Monitor Performance Monitor Events

The WM can produce a performance monitor (PM) event with every trigger. This is accomplished by configuring the performance monitor to count WM events. For more information on this configuration see the events named 'Number of watchpoint monitor hits' and 'Number of trace buffer hits' in Table 19-10.

Multi-level triggers can be created using the watchpoint monitor, the performance monitor, and the trace buffer combined. For example, the WM can be programmed to trigger on events that also increment a PM counter (the performance monitor must also be programmed to respond to this event), the output of which (perfmon_overflow) could trigger the start of tracing in the trace buffer.

20.4.6 Trace Buffer

The trace buffer is a 256 × 64 array that can capture information about the internal processing of transactions to selected interfaces. The trace buffer controls are a superset of those for the watchpoint monitor. Close inspection of the trace buffer control registers (TBCR*n*) and the WM control registers (WMCR*n*) shows that trace buffer controls not needed for the WM are marked reserved in WMCR*n*. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer provides great flexibility about when to start tracing, when to stop tracing, and what to trace. The trace mode field, TBCR0[MODE], indicates when to trace: on every valid cycle, on a watchpoint monitor event, or when all the programmed events in the TBCR are met. This permits a user to program the trace condition in the watchpoint monitor and to program a start or stop condition in the trace buffer control register. The user can also program the TBCR with the conditions in which to stop tracing: on an event, or when the buffer is full. TBCR0[IFSEL] specifies which interface transactions are being captured.

The trace buffer can be programmed to trace the dispatch bus from any of the following:

- e500 coherency module (ECM)
- Outbound host interface to the PCI controller
- Host interface to the DDR controller

Transactions come into the ECM, arbitrate for common resources, and get dispatched to the target port. Information such as transaction types, source ID, and other attributes can be captured in any of the selected interfaces.

20.4.6.1 Traced Data Formats (as a Function of TBCR1[IFSEL])

Figure 20-20 shows the trace buffer entry format for an ECM dispatch (CMD) transaction that is specified when TBCR1[IFSEL] = 000.

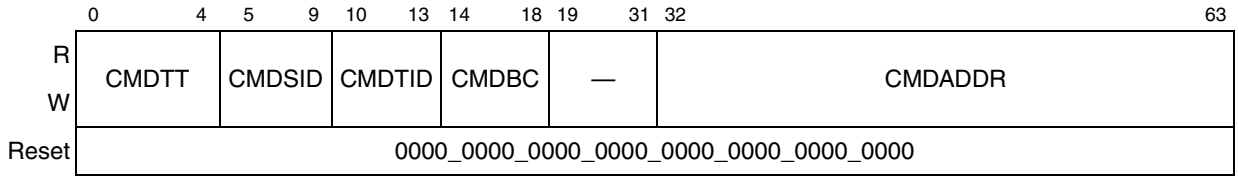


Figure 20-20. e500 Coherency Module Dispatch (CMD) Trace Buffer Entry

Table 20-27 describes the fields of CMD trace buffer entries.

Table 20-27. CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000)

Bits	Name	Function
0–4	CMDTT	Transaction type. Specifies the transaction type as shown in Table 20-12. For example, a value of zero indicates a write with local processor snoop condition.
5–9	CMDSID	Source ID. Identifies the source of the transaction as shown in Table 20-26. For example, a value of 010101 indicates that DMA is the transaction source.
10–13	CMDTID	Target ID. Identifies the target of the transaction as shown in Table 20-26. For example, a value of 010101 indicates that DMA is the transaction target.
14–18	CMDBC	Byte count. Range: 32 to 1 where a value of 0 indicates 32 bytes. 00000 = 32 bytes 00001 = 1 byte 00010 = 2 bytes . . . 11110 = 30 bytes 11111 = 31 bytes
19–31	—	Reserved
32–63	CMDADDR	Address bits 0–31

Figure 20-21 shows the trace buffer entry format for the DDR SDRAM interface, TBCR1[IFSEL] = 001.

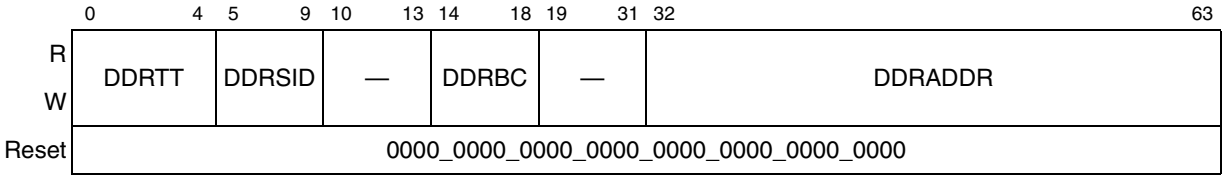


Figure 20-21. DDR Trace Buffer Entry

Table 20-28 describes the fields of DDR SDRAM trace buffer entries when TBCR1[IFSEL] = 001.

Table 20-28. DDR Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 001)

Bits	Name	Function
0–4	DDRTT	Transaction type. Specifies the transaction type as shown in Table 20-12. For example, a value of all zeros maps to write.
5–9	DDRSID	Source ID. Specifies the source of the transaction as shown in Table 20-26. For example, a value of 010101 indicates that DMA is the transaction source, and so on.
10–13	—	Reserved
14–18	DDRBC	Byte count
19–31	—	Reserved
32–63	DDRADDR	Address bits 0–31

Figure 20-22 shows the PCI trace buffer entry format when TBCR1[IFSEL] = 010.

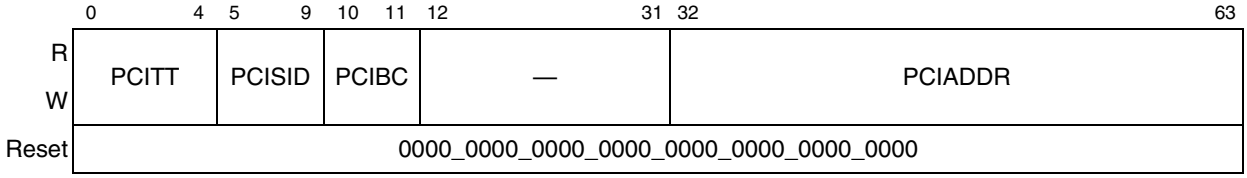


Figure 20-22. PCI Trace Buffer Entry

Table 20-29 describes the fields of PCI trace buffer entries when TBCR1[IFSEL] = 010.

Table 20-29. PCI Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 010)

Bits	Name	Function
0–4	PCITT	Transaction type. Specifies the transaction type as shown in Table 20-12. For example, a value of all zeros maps to write.
5–9	PCISID	Source ID. Identifies the source of the transaction as shown in Table 20-26. For example, a value of 010101 identifies DMA as the transaction source.
10–11	PCIBC	Byte count. The size of the transaction. 00 32 bytes 01 8 bytes 10 16 bytes 11 24 bytes
12–31	—	Reserved
32–63	PCIADDR	Address bits 0–31

20.5 Initialization

Configuring the appropriate control register must be the last step in the initialization sequence for either the watchpoint or trace buffer. That is, all required registers except the corresponding control register must be configured before any control register bits that enable watchpoint or trace events are set.

Part V
CPM Features

The MPC8555E communications processor module (CPM) is a superset of the MPC8260 PowerQUICC II CPM, with enhancements in performance and the addition of hardware and microcode routines for supporting high bit-rate protocols like ATM and fast Ethernet. The support for multiple high-level data link control (HDLC) channels is enhanced to support up to 256 HDLC channels. This part defines the CPM blocks of the MPC8555E. It contains the following chapters:

- Chapter 21, "Communications Processor Module Overview," provides a high-level summary of the MPC8555E features and memory map.
- Chapter 22, "CPM Interrupt Controller," describes the CPM interrupt controller of the MPC8555E.
- Chapter 23, "Serial Interface with Time-Slot Assigner," describes the serial interface and TSA of the MPC8555E.
- Chapter 24, "CPM Multiplexing," describes how the CPM multiplexing logic (CMX) connects the physical layer (UTOPIA, MII, modem lines, TDM lines, and proprietary serial lines) to the FCCs and SCCs.
- Chapter 25, "Baud-Rate Generators (BRGs)," describes the eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs and SCCs.
- Chapter 26, "CPM Timers," describes the four identical 16-bit general-purpose CPM timers that can alternately be used as two 32-bit timers.
- Chapter 27, "SDMA Channels," describes the two physical serial DMA (SDMA) channels of the MPC8555E.
- Chapter 28, "Serial Communications Controllers (SCCs)," describes the MPC8555E's four SCCs, which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks.
- Chapter 29, "SCC UART Mode," describes how the general SCC mode register (GSMR) is used to configure an SCC channel to function in UART mode, which provides standard serial I/O using asynchronous character-based (start-stop) protocols with RS-232C-type lines.
- Chapter 30, "SCC HDLC Mode," describes how HDLC mode is selected for an SCC. In HDLC mode, an SCC becomes an HDLC controller, and consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to other SCCs.
- Chapter 31, "SCC BISYNC Mode," describes how transparent BISYNC mode allows full binary data to be sent with any possible character pattern.
- Chapter 32, "SCC Transparent Mode," describes how an SCC in transparent mode functions as a high-speed serial-to-parallel and parallel-to-serial converter.

- Chapter 33, “SCC AppleTalk Mode,” describes how the MPC8555E provides LocalTalk protocol support. The AppleTalk controller provides required frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames.
- Chapter 34, “QUICC Multi-Channel Controller (QMC),” describes the QMC protocol, which emulates up to 64 logical channels within one SCC using the same time-division-multiplexed (TDM) physical interface.
- Chapter 35, “Universal Serial Bus Controller,” describes the MPC8555E USB controller, including basic operation, the parameter RAM, and registers.
- Chapter 36, “Serial Management Controllers (SMCs),” describes two serial management controllers, full-duplex ports that can be configured independently to support one of three protocols—UART, transparent, or general-circuit interface (GCI).
- Chapter 37, “Fast Communications Controllers (FCCs),” describes how the FCCs can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks.
- Chapter 38, “FCC HDLC Controller,” describes the FCC HDLC controller of the MPC8555E.
- Chapter 39, “FCC Transparent Controller,” describes how the FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter.
- Chapter 40, “CPM Fast Ethernet Controller,” describes the fast Ethernet controller in the CPM.
- Chapter 41, “ATM Controller,” describes the ATM controller that provides the ATM and AAL layers of the ATM protocol using the universal test and operations physical layer (PHY) interface for ATM (UTOPIA level II) for both master and slave modes.
- Chapter 42, “ATM AAL2,” describes the AAL2 microcode package.
- Chapter 43, “Serial Peripheral Interface (SPI),” describes the serial peripheral interface (SPI) of the MPC8555E CPM.
- Chapter 44, “I^2C Controller,” describes the I^2C controller of the CPM.
- Chapter 45, “Parallel I/O Ports,” describes the four general-purpose I/O ports of the CPM.

Convention

This part uses the following additional notational convention:

Bold	In figures and tables showing registers and parameter RAM, bold entries indicate fields that should be initialized by the user.

Chapter 21
Communications Processor Module Overview

The MPC8555E communications processor module (CPM) is a modified version of the MPC8260 PowerQUICC II CPM, with enhancements in performance and the addition of hardware and microcode routines for supporting high bit-rate protocols like ATM and Fast Ethernet. The support for multiple high-level data link control (HDLC) channels is enhanced to support up to 64 HDLC channels.

21.1 Features

The CPM includes various blocks to provide the system with an efficient way to handle data communication tasks. The following is a list of the CPM's important features.

- Communications processor (CP)
  - — One instruction per clock
  - — Executes code from internal ROM or instruction RAM
  - — 32-bit RISC architecture
  - — Tuned for communication environments: instruction set supports CRC computation and bit manipulation.
  - — Internal timer
  - — Interfaces with the embedded e500 core processor through dual-port RAM and virtual DMA channels for each peripheral controller. (Dual-port RAM size is 16 Kbytes plus 4 Kbytes of dedicated instruction RAM.)
  - — Handles serial protocols and virtual DMA
- Two full-duplex fast serial communications controllers (FCCs) support the following protocols:
  - — ATM protocol through UTOPIA interface
  - — IEEE 802.3 standard/Fast Ethernet
  - — High level data link control (HDLC)
  - — Totally transparent operation
- Three full-duplex serial communications controllers (SCCs) support the following protocols:
  - — High level/synchronous data link control (HDLC/SDLC)
  - — LocalTalk (HDLC-based local area network protocol)
  - — Universal asynchronous receiver transmitter (UART)
  - — Synchronous UART (1× clock mode)
  - — Binary synchronous communication (BISYNC)
  - — Totally transparent operation
  - — QMC support, providing 64 channels with only one TDM interface

- Two full-duplex serial management controllers (SMCs) support the following protocols:
  - GCI (ISDN interface) monitor and C/I channels
  - UART
  - Totally transparent operation
- Serial peripheral interface (SPI) support for master or slave
- I^2C bus controller
- Time-slot assigner supports multiplexing of data from any of the SCCs, FCCs, and SMCs. The time-slot assigner supports the following TDM formats:
  - T1/CEPT lines
  - T3/E3
  - Pulse code modulation (PCM) highway interface
  - ISDN primary rate
  - Freescale interchip digital link (IDL)
  - General circuit interface (GCI)
  - User-defined interfaces
- Universal serial bus (USB 2.0)
- Eight independent baud rate generators (BRGs)
- Four general-purpose 16-bit timers or two 32-bit timers
- General-purpose parallel ports—sixteen parallel I/O lines with interrupt capability

Figure 21-1 shows the MPC8555E CPM block diagram.

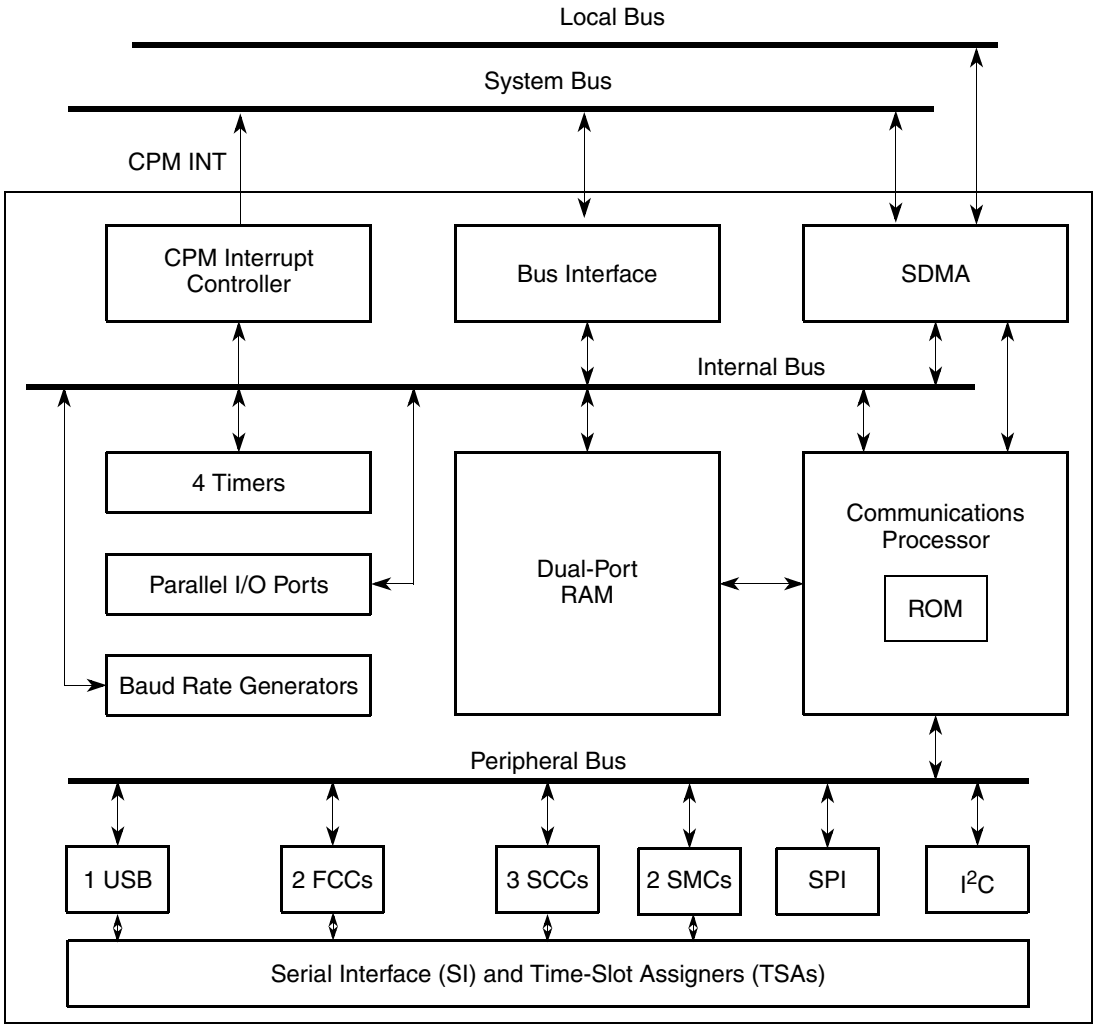


Figure 21-1. MPC8555E CPM Block Diagram

NOTE

The CPM clock frequency is the same as the CCB clock frequency and is determined by the configuration of the platform PLL during power-on reset.

21.1.1 CPM Memory Map

Table 21-1 shows the CPM portion of the internal memory map.

Table 21-1. MPC8555E Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
CPM Dual-Port RAM				
0x8_0000–0x8_1FFF	DPRAM1—Dual-port RAM	R/W	—	21.4/21-28
0x8_2000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_9FFF	DPRAM2—Dual-port RAM	R/W	—	21.4/21-28
0x8_A000–0x8_FFFF	Reserved	—	—	—
e500 Core Interface				
0x9_0000	CEAR—CPM error address register	R	0x0000_0000	21.2.3.1.1/21-18
0x9_0004	CEER—CPM error event register	R/W	0x0000	21.2.3.1.2/21-19
0x9_0006	CEMR—CPM error mask register	R/W	0x0000	21.2.3.1.3/21-20
SDMA				
0x9_0050	SMAER—System bus address error register	R	0x0000_0000	27.1.1/27-2
0x9_0054	Reserved	—	—	—
0x9_0058	SMEVR—System bus event register	R/W	0x0000_0000	27.1.2/27-2
0x9_005C	SMCTR—System bus control register	R/W	0x3800_0000	27.1.3/27-3
0x9_0060	LMAER—Local bus address error register	R	0x0000_0000	27.1.1/27-2
0x9_0064	Reserved	—	—	—
0x9_0068	LMEVR—Local bus event register	R/W	0x0000_0000	27.1.2/27-2
0x9_006C	LMCTR—Local bus control register	R/W	0x3800_0000	27.1.3/27-3
Interrupt Controller				
0x9_0C00	SICR—CPM interrupt configuration register	R/W	0x0000_0000	22.5.1.1/22-9
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIVEC—CPM interrupt vector register	R/W	0x0000_0000	22.5.1.5/22-14
0x9_0C08	SIPNR_H—CPM interrupt pending register (high)	R/W	0x0000_0000	22.5.1.3/22-11
0x9_0C0C	SIPNR_L—CPM interrupt pending register (low)	R/W	0x0000_0000	22.5.1.3/22-11
0x9_0C10	Reserved	—	—	—
0x9_0C14	SCPRR_H—CPM interrupt priority register (high)	R/W	0x0530_9770	22.5.1.2/22-10

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_0C18	SCPRR_L—CPM interrupt priority register (low)	R/W	0x0530_9770	22.5.1.2/22-10
0x9_0C1C	SIMR_H—CPM interrupt mask register (high)	R/W	0x0000_0000	22.5.1.4/22-12
0x9_0C20	SIMR_L—CPM interrupt mask register (low)	R/W	0x0000_0000	22.5.1.4/22-12
0x9_0C24	SIEXR—CPM external interrupt control register	R/W	0x0000_0000	22.5.1.6/22-15
0x9_0C28–0x9_0C7F	Reserved	—	—	—
Clock				
0x9_0C80	SCCR—System clock control register	R/W	0x0000_0000	25.1/25-2
Input/Output Port				
0x9_0D00	PDIRA—Port A data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D04	PPARA—Port A pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D08	PSORA—Port A special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D0C	PODRA—Port A open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D10	PDATA—Port A data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D14–0x9_0D1F	Reserved	—	—	—
0x9_0D20	PDIRB—Port B data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D24	PPARB—Port B pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D28	PSORB—Port B special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D2C	PODRB—Port B open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D30	PDATB—Port B data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	PDIRC—Port C data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D44	PPARC—Port C pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D48	PSORC—Port C special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D4C	PODRC—Port C open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D50	PDATC—Port C data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	PDIRD—Port D data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D64	PPARD—Port D pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D68	PSORD—Port D special options register	R/W	0x0000_0000	45.2.5/45-13

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_0D6C	PODRD—Port D open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D70	PDATD—Port D data register	R/W	0x0000_0000	45.2.2/45-4
CPM Timers				
0x9_0D80	TGCR1—Timer 1 and timer 2 global configuration register	R/W	0x00	26.2.2/26-3
0x9_0D81	Reserved	—	—	—
0x9_0D84	TGCR2—Timer 3 and timer 4 global configuration register	R/W	0x00	26.2.2/26-3
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	TMR1—Timer 1 mode register	R/W	0x0000	26.2.3/26-5
0x9_0D92	TMR2—Timer 2 mode register	R/W	0x0000	26.2.3/26-5
0x9_0D94	TRR1—Timer 1 reference register	R/W	0x0000	26.2.4/26-7
0x9_0D96	TRR2—Timer 2 reference register	R/W	0x0000	26.2.4/26-7
0x9_0D98	TCR1—Timer 1 capture register	R/W	0x0000	26.2.5/26-7
0x9_0D9A	TCR2—Timer 2 capture register	R/W	0x0000	26.2.5/26-7
0x9_0D9C	TCN1—Timer 1 counter	R/W	0x0000	26.2.6/26-7
0x9_0D9E	TCN2—Timer 2 counter	R/W	0x0000	26.2.6/26-7
0x9_0DA0	TMR3—Timer 3 mode register	R/W	0x0000	26.2.3/26-5
0x9_0DA2	TMR4—Timer 4 mode register	R/W	0x0000	26.2.3/26-5
0x9_0DA4	TRR3—Timer 3 reference register	R/W	0x0000	26.2.4/26-7
0x9_0DA6	TRR4—Timer 4 reference register	R/W	0x0000	26.2.4/26-7
0x9_0DA8	TCR3—Timer 3 capture register	R/W	0x0000	26.2.5/26-7
0x9_0DAA	TCR4—Timer 4 capture register	R/W	0x0000	26.2.5/26-7
0x9_0DAC	TCN3—Timer 3 counter	R/W	0x0000	26.2.6/26-7
0x9_0DAE	TCN4—Timer 4 counter	R/W	0x0000	26.2.6/26-7
0x9_0DB0	TER1—Timer 1 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB2	TER2—Timer 2 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB4	TER3—Timer 3 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB6	TER4—Timer 4 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB8–0x9_12FF	Reserved	—	—	—

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC1				
0x9_1300	GFMR1—FCC1 general mode register	R/W	0x0000_0000	37.2/37-3
0x9_1304	FPSMR1—FCC1 protocol-specific mode register	R/W	0x0000_0000	(ATM) 41.13.3/41-84 (Ethernet) 40.18.2/40-20 (HDLC) 38.6/38-8
0x9_1308	FTODR1—FCC1 transmit on demand register	R/W	0x0000	37.6/37-8
0x9_130A	Reserved	—	—	—
0x9_130C	FDSR1—FCC1 data synchronization register	R/W	0x7E7E	37.5/37-7
0x9_130E	Reserved	—	—	—
0x9_1310	FCCE1—FCC1 event register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1314	FCCM1—FCC1 mask register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1316	Reserved	—	—	—
0x9_1318	FCCS1—FCC1 status register	R	0x00	38.10/38-16 (HDLC)
0x9_1319	Reserved	—	—	—
0x9_131C	FTIRR1_PHY0—FCC1 transmit internal rate registers for PHY0	R/W	0x00	41.13.5/41-87 (ATM)
0x9_131D	FTIRR1_PHY1—FCC1 transmit internal rate registers for PHY1	R/W	0x00	41.13.5/41-87 (ATM)
0x9_131E	FTIRR1_PHY2—FCC1 transmit internal rate registers for PHY2	R/W	0x00	41.13.5/41-87 (ATM)
0x9_131F	FTIRR1_PHY3—FCC1 transmit internal rate registers for PHY3	R/W	0x00	41.13.5/41-87 (ATM)

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC2				
0x9_1320	GFMR2—FCC2 general mode register	R/W	0x0000_0000	37.2/37-3
0x9_1324	FPSMR2—FCC2 protocol-specific mode register	R/W	0x0000_0000	(ATM) 41.13.3/41-84 (Ethernet) 40.18.2/40-20 (HDLC) 38.6/38-8
0x9_1328	FTODR2—FCC2 transmit on-demand register	R/W	0x0000	37.6/37-8
0x9_132A	Reserved	—	—	—
0x9_132C	FDSR2—FCC2 data synchronization register	R/W	0x7E7E	37.5/37-7
0x9_132E	Reserved	—	—	—
0x9_1330	FCCE2—FCC2 event register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1334	FCCM2—FCC2 mask register	R/W	0x0000_0000	(ATM) 41.13.4/41-86 (Ethernet) 40.18.3/40-22 (HDLC) 38.9/38-14
0x9_1336	Reserved	—	—	—
0x9_1338	FCCS2—FCC2 status register	R	0x00	38.10/38-16 (HDLC)
0x9_1339–0x9_137F	Reserved	—	0x00	—
0x9_133C	FTIRR2_PHY0—FCC2 transmit internal rate registers for PHY0	R/W	0x00	41.13.5/41-87 (ATM)
0x9_133D	FTIRR2_PHY1—FCC2 transmit internal rate registers for PHY1	R/W	0x00	41.13.5/41-87 (ATM)
0x9_133E	FTIRR2_PHY2—FCC2 transmit internal rate registers for PHY2	R/W	0x00	41.13.5/41-87 (ATM)
0x9_133F	FTIRR2_PHY3—FCC2 transmit internal rate registers for PHY3	R/W	0x00	41.13.5/41-87 (ATM)
0x9_1340–0x9_137F	Reserved	—	—	—

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC1 (continued)				
0x9_1380	FIRPER1—FCC1 internal rate port enable register	R/W	0x0000_0000	41.15.3/41-90
0x9_1384	FIRER1—FCC1 internal rate event register	R/W	0x0000_0000	41.15.4/41-91
0x9_1388	FIRSR1_HI—FCC1 internal rate selection register:HI	R/W	0x0000_0000	41.15.5/41-92
0x9_138C	FIRSR1_LO— FCC1 internal rate selection register:LO	R/W	0x0000_0000	41.15.5/41-92
0x9_1390	GFEMR1—General FCC1 expansion mode register	R/W	0x00	37.3/37-7
0x9_1391–0x9_139F	Reserved	—	—	—
FCC2 (continued)				
0x9_13A0	FIRPER2—FCC2 internal rate port enable register	R/W	0x0000_0000	41.15.3/41-90
0x9_13A4	FIRER2—FCC2 internal rate event register	R/W	0x0000_0000	41.15.4/41-91
0x9_13A8	FIRSR2_HI—FCC2 internal rate selection register:HI	R/W	0x0000_0000	41.15.5/41-92
0x9_13AC	FIRSR2_LO—FCC2 internal rate selection register:LO	R/W	0x0000_0000	41.15.5/41-92
0x9_13B0	GFEMR2—General FCC2 expansion mode register	R/W	0x00	37.3/37-7
0x9_13B1–0x9_15EF	Reserved	—	—	—
BRGs 5–8				
0x9_15F0	BRGC5—BRG5 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15F4	BRGC6—BRG6 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15F8	BRGC7—BRG7 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15FC	BRGC8—BRG8 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_1600–0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I2MOD—I^2C mode register	R/W	0x00	44.4.1/44-6
0x9_1861	Reserved	—	—	—
0x9_1864	I2ADD—I^2C address register	R/W	0x00	44.4.2/44-7
0x9_1865	Reserved	—	—	—
0x9_1868	II2BRG—I^2C BRG register	R/W	0x00	44.4.3/44-7
0x9_1869	Reserved	—	—	—
0x9_186C	I2COM—I^2C command register	R/W	0x00	44.4.5/44-8
0x9_186D	Reserved	—	—	—

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1870	I2CER—I^2C event register	R/W	0x00	44.4.4/44-7
0x9_1871	Reserved	—	—	—
0x9_1874	I2CMR—I^2C mask register	R/W	0x00	44.4.4/44-7
0x9_1875–0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	CPCR—Communications processor command register	R/W	0x0000_0000	21.3.1/21-24
0x9_19C4	RCCR—CP configuration register	R/W	0x0000_0000	21.2.6/21-22
0x9_19C8–0x9_19D5	Reserved	—	—	—
0x9_19D6	RTER—CP timers event register	R/W	0x0000	21.5.4/21-35
0x9_19DA	RTMR—CP timers mask register	R/W	0x0000	21.5.4/21-35
0x9_19DC	RTSCR—CP time-stamp timer control register	R/W	0x0000	21.2.7/21-23
0x9_19DE	Reserved	—	—	—
0x9_19E0	RTSR—CP time-stamp register	R/W	0x0000_0000	21.2.8/21-24
BRGs 1–4				
0x9_19F0	BRGC1—BRG1 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19F4	BRGC2—BRG2 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19F8	BRGC3—BRG3 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19FC	BRGC4—BRG4 configuration register	R/W	0x0000_0000	25.2/25-3
SCC1				
0x9_1A00	GSMR_L1—SCC1 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A04	GSMR_H1—SCC1 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A08	PSMR1—SCC1 protocol-specific mode register	R/W	0x0000	28.2/28-3 29.16/29-12 (UART) 30.8/30-7 (HDLC) 31.11/31-9 (BISYNC) 32.9/32-8 (Transparent)
0x9_1A0A	Reserved	—	—	—
0x9_1A0C	TODR1—SCC1 transmit-on-demand register	R/W	0x0000	28.2.3/28-9
0x9_1A0E	DSR1—SCC1 data synchronization register	R/W	0x7E7E	28.2.2/28-8

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A10	SCCE1—SCC1 event register	R/W	0x0000	29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A12–0x9_1A13	Reserved	—	—	—
0x9_1A14	SCCM1—SCC1 mask register	R/W	0x0000	29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A16	Reserved	—	—	—
0x9_1A17	SCCS1—SCC1 status register	R/W	0x00	29.20/29-20 (UART) 30.12/30-13 (HDLC) 31.15/31-15 (BISYNC) 32.13/32-12 (Transparent)
0x9_1A18–0x9_1A3F	Reserved	—	—	—
SCC3				
0x9_1A40	GSMR_L3—SCC3 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A44	GSMR_H3—SCC3 general mode register	R/W	0x0000_0000	
0x9_1A48	PSMR3—SCC3 protocol-specific mode register	R/W	0x0000	28.2.1/28-8 29.16/29-12 (UART) 30.8/30-7 (HDLC) 31.11/31-9 (BISYNC) 32.9/32-8 (Transparent)
0x9_1A4A	Reserved	—	—	—
0x9_1A4C	TODR3—SCC3 transmit on demand register	R/W	0x0000	28.2.2/28-8
0x9_1A4E	DSR3—SCC3 data synchronization register	R/W	0x7E7E	28.2.2/28-8

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A50	SCCE3—SCC3 event register	R/W		29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A52–0x0x9_1A53	Reserved	—	—	—
0x9_1A54	SCCM3—SCC3 mask register	R/W	0x0000	29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A56	Reserved	—	—	—
0x9_1A57	SCCS3—SCC3 status register	R/W	0x00	29.20/29-20 (UART) 30.12/30-13 (HDLC) 31.15/31-15 (BISYNC) 32.13/32-12 (Transparent)
0x9_1A58–0x9_1A5F	Reserved	—	—	—
SCC4				
0x9_1A60	GSMR_L4—SCC4 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A64	GSMR_H4—SCC4 general mode register	R/W	0x0000_0000	28.2/28-3
0x9_1A68	PSMR4—SCC4 protocol-specific mode register	R/W	0x0000	28.2.1/28-8 29.16/29-12 (UART) 30.8/30-7 (HDLC) 31.11/31-9 (BISYNC) 32.9/32-8 (Transparent)
0x9_1A6A	Reserved	—	—	—
0x9_1A6C	TODR4—SCC4 transmit on-demand register	R/W	0x0000	28.2.3/28-9

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A6E	DSR4—SCC4 data synchronization register	R/W	0x7E7E	28.2.2/28-8
0x9_1A70	SCCE4—SCC4 event register	R/W		29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A72–0x9_1A73	Reserved	—	—	—
0x9_1A74	SCCM4—SCC4 mask register	R/W	0x0000	29.19/29-18 (UART) 30.11/30-12 (HDLC) 31.14/31-14 (BISYNC) 32.12/32-11 (Transparent)
0x9_1A76	Reserved	—	—	—
0x9_1A77	SCCS4—SCC4 status register	—	—	29.20/29-20 (UART) 30.12/30-13 (HDLC) 31.15/31-15 (BISYNC) 32.13/32-12 (Transparent)
0x9_1A78–0x9_1A7F	Reserved	—	—	—
SMC1				
0x9_1A82	SMCMR1—SMC1 mode register	R/W	0x0000	36.2.1/36-2
0x9_1A84	Reserved	—	16 bits	—
0x9_1A86	SMCE1—SMC1 event register	R/W	0x00	36.3.11/36-18
0x9_1A87	Reserved	—	24 bits	—
0x9_1A8A	SMCM1—SMC1 mask register	R/W	0x00	36.3.11/36-18
0x9_1A8B–0x9_1A91	Reserved	—	7 bytes	—
SMC2				
0x9_1A92	SMCMR2—SMC2 mode register	R/W	0x0000	36.2.1/36-2
0x9_1A94	Reserved	—	16 bits	—

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A96	SMCE2—SMC2 event register	R/W	0x00	36.3.11/36-18
0x9_1A97	Reserved	—	24 bits	—
0x9_1A9A	SMCM2—SMC2 mask register	R/W	0x00	36.3.11/36-18
0x9_1A9B–0x9_1A9F	Reserved	—	5 bytes	—
SPI				
0x9_1AA0	SPMODE—SPI mode register	R/W	0x0000	43.4.1/43-6
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPIE—SPI event register	R/W	0x00	43.4.2/43-9
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPIM—SPI mask register	R/W	0x00	43.4.2/43-9
0x9_1AAB	Reserved	—	—	—
0x9_1AAD	SPCOM—SPI command register	W	0x00	43.4.3/43-10
0x9_1AA7–0x9_1B00	Reserved	—	—	—
CPM Mux				
0x9_1B02	CMXSI2CR—CPM mux SI2 clock route register	R/W	0x00	24.4.2/24-8
0x9_1B03	Reserved	—	—	—
0x9_1B04	CMXFCR—CPM mux FCC clock route register	R/W	0x0000_0000	24.4.3/24-8
0x9_1B08	CMXSCR—CPM mux SCC clock route register	R/W	0x0000_0000	24.4.4/24-10
0x9_1B0C	CMXSMR—CPM mux SMC clock route register	R/W	0x00	24.4.5/24-13
0x9_1B0E	CMXUAR—CPM mux UTOPIA address register	R/W	0x0000	24.4.1/24-5
0x9_1B10–0x9_1B3F	Reserved	—	—	—
SI2 Registers				
0x9_1B40	SI2AMR—SI2 TDMA2 mode register	R/W	0x0000	23.6.2/23-14
0x9_1B42	SI2BMR—SI2 TDMB2 mode register	R/W	0x0000	23.6.2/23-14
0x9_1B44	SI2CMR—SI2 TDMC2 mode register	R/W	0x0000	23.6.2/23-14
0x9_1B46	Reserved	R/W	16 bits	23.6.2/23-14
0x9_1B48	SI2GMR—SI2 global mode register	R/W	0x00	23.6.1/23-14
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2CMDR—SI2 command register	R/W	0x00	23.6.4/23-20
0x9_1B4B	Reserved	—	—	—

Table 21-1. MPC8555E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1B4C	SI2STR—SI2 status register	R/W	0x00	23.6.5/23-21
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2RSR—SI2 RAM shadow address register	R/W	0x0000	23.6.3/23-20
0x9_1B50–0x9_1B5F	Reserved	—	—	—
USB				
0x9_1B60	USMOD—USB mode register	R/W	0x00	35.5.7.1/35-17
0x9_1B61	USADR—USB address register	R/W	0x00	35.5.7.2/35-18
0x9_1B62	USCOM—USB command register	R/W	0x00	35.5.7.4/35-19
0x9_1B64	USEP1—USB endpoint 1 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B66	USEP2—USB endpoint 2 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B68	USEP3—USB endpoint 3 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B6A	USEP4—USB endpoint 4 register	R/W	0x0000	35.5.7.3/35-18
0x9_1B6C–0x9_1B6F	Reserved	—	32 bits	—
0x9_1B70	USBER—USB event register	R/W	0x0000	35.5.7.5/35-20
0x9_1B72	Reserved	—	16 bits	—
0x9_1B74	USBMR—USB mask register	R/W	0x0000	35.5.7.6/35-21
0x9_1B77	USBS—USB status register	R/W	0x00	35.5.7.7/35-21
0x9_1B79–0x9_1FFF	Reserved	—	1174 bytes	—
SI2 RAM				
0x9_2800–0x9_29FF	SI2TxRAM—SI 2 transmit routing RAM	—	—	23.5.3/23-9
0x9_2A00–0x9_2BFF	Reserved	—	—	—
0x9_2C00–0x9_2DFF	SI2RxRAM—SI 2 receive routing RAM	—	—	23.5.3/23-9
0x9_2E00–0x9_3FFF	Reserved	—	—	—
Instruction RAM				
0xA_0000–0xA_0FFF	Dual-port RAM (instruction RAM only)	—	—	21.4/21-28

21.2 Communications Processor (CP)

The communications processor (CP), also called the RISC microcontroller, is a 32-bit controller for the CPM that resides on a separate bus from the core and, therefore, can perform tasks independent of the e500 core. The CP handles lower-layer communications tasks and DMA control, freeing the core to handle higher-layer activities. The CP works with the peripheral controllers and parallel port to implement user-programmable protocols and manage the serial DMA (SDMA) channels that transfer data between the I/O channels and memory. It also contains an internal timer used to implement up to 16 additional software timers.

The CP's architecture and instruction set are optimized for data communications and data processing required by many wire-line and wireless communications standards.

21.2.1 Features

The following is a list of the CP's primary features.

- One system clock cycle per instruction
- 32-bit instruction object code
- Executes code from internal ROM or instruction RAM
- 32-bit ALU data path
- 64-bit internal RAM access
- Optimized for communications processing
- Performs DMA bursting of serial data from/to internal RAM/external memory
- Tuned for communications environments—instruction set supports CRC computation and bit manipulation
- Internal timer
- Interfaces with the CPU through 16 Kbytes of internal dual-port RAM and virtual DMA channels for each serial channel
- Handles serial protocols

21.2.2 CP Block Diagram

The CP contains the following functional units:

- Scheduler and sequencer
- Instruction decoder
- Execution unit
- Load/store unit (LSU)
- Block transfer module (BTM)—moves data between serial FIFO and RAM
- Eight general purpose registers (GPRs)
- Special registers, CRC machine, HDLC framer

The CP also gives SDMA commands to the SDMA. The CP interfaces with the dual-port RAM for loading and storing data.

Figure 21-2 shows the CP block diagram.

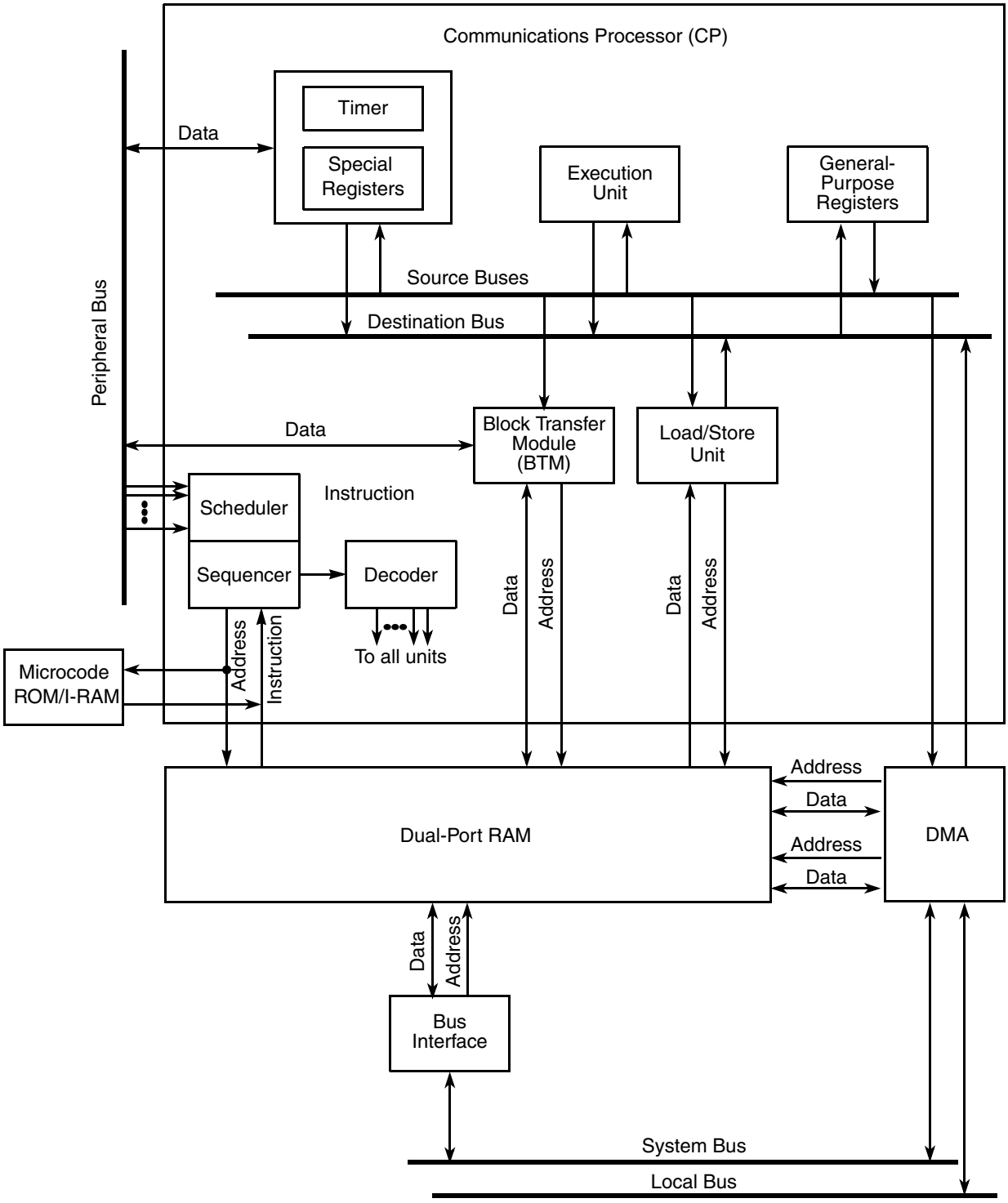


Figure 21-2. Communications Processor (CP) Block Diagram

21.2.3 e500 Core Interface

The CP communicates with the e500 core in several ways:

- Many parameters are exchanged through the dual-port RAM.
- The CP can execute special commands issued by the core. These commands should only be issued in special situations like exceptions or error recovery.
- The CP generates interrupts to the CPM interrupt controller, which is connected to the programmable interrupt controller (PIC) of the MPC8555E.
- The e500 core can read the CPM status/event registers at any time.

21.2.3.1 Error Reporting and Capture

When the e500 core (or any other master) tries to read from or write to the CPM registers or internal memories and an erroneous condition is recognized, the CPM reports the event and captures the information in the CPM error registers. The following sections describe these registers.

21.2.3.1.1 CPM Error Address Register (CEAR)

The CPM error address register (CEAR), shown in Figure 21-3, is a read-only register that contains the address and attributes of the first erroneous transaction that occurred after the last time that the CPM error event register (CEER) was entirely cleared. Note that this register contains invalid data if the CPM error event register has all of its bits cleared. The CEER is cleared by $\overline{\text{HRESET}}$ or as a result of a write transaction.

CEAR is not affected by write transactions.

	0	7	8	9	10	15
Field	BYTE_EN		RW	—	ADDR	
R/W	R					
Reset:	0000_0000_0000_0000[1]					
Offset	0x90000					

	16	31
Field	ADDR	
R/W	R	
Reset	0000_0000_0000_0000[1]	
Offset	0x90002	

[1] CEAR is unaffected by soft reset. It is cleared by hard reset.

Figure 21-3. CPM Error Address Register (CEAR)

Table 21-2. CEAR Field Descriptions

Bits	Name	Description
0–7	BYTE_EN	Byte enables. Contains the byte enable pattern of the erroneous transaction
8	RW	Read/Write. The read/write attribute of the erroneous transaction 1 Erroneous transaction type is read 0 Erroneous transaction type is write
9	—	Reserved.
10–31	ADDR	Address. The address of the erroneous transaction. ADDR[29–31] are always read as zeros. This field contains the internal CPM offset, while the full address is a combination of this field and the base address of the CPM as indicated by the CCSRBAR setting.

21.2.3.1.2 CPM Error Event Register (CEER)

The CPM error event register contains the type of error if an error event has occurred. When an erroneous transaction is recognized, the CPM sets the corresponding bit in CEER. An interrupt is generated if enabled by the CPM error mask register. Note that CEER reflects one or more erroneous events.

CEER bits are cleared by $\overline{\text{HRESET}}$ or by writing 1 to the appropriate bit; writing 0 has no effect.

	0	1	2	3	4 … 15
Field	ABORT	BE	SIZE	NOMAP	—
Reset	0000_0000_0000_0000[1]				
R/W	R/W				
Offset	0x90004				

[1] CEER is unaffected by soft reset. It is cleared by hard reset.

Figure 21-4. CPM Error Event Register (CEER)

Table 21-3. CEER Field Descriptions

Bits	Name	Description
0	ABORT	Abort error. Asserted if a transaction that was targeted to the CPM was aborted before completion due to an internal error condition in the MPC8555E
1	BE	Byte enable error. Set when a transaction with a non-contiguous or all-zeros byte enable pattern is targeted to the CPM
2	SIZE	Size error. Set when an access is attempted to CPM registers, with a byte enable pattern that crosses an aligned 4-byte boundary
3	NOMAP	Not mapped. Set when an access is targeted into a non-implemented address space. None implemented space is defined by: CPM Base + 0x14000 <= Address <= CPM Base + 0x1FFFF, or CPM Base + 0x30000 <= Address <= CPM Base + 0x3FFFF
4–15	—	Reserved, should be cleared.

21.2.3.1.3 CPM Error Mask Register (CEMR)

The CPM error mask register (CEMR) is used to mask corresponding event bits in CEER. Setting a mask bit enables the corresponding CEER interrupt.

	0	1	2	3	4	15
Field	ABORT	BE	SIZE	NOMAP	—	
Reset	0000_0000_0000_0000					
R/W	R/W					
Offset	0x90006					

Figure 21-5. CPM Error Mask Register (CEMR)

Table 21-4. CEMR Field Descriptions

Bits	Name	Description
0	ABORT	Mask interrupt for ABORT event 0 Interrupt is disabled 1 Interrupt is enabled
1	BE	Mask interrupt for BE error event 0 Interrupt is disabled 1 Interrupt is enabled
2	SIZE	Mask interrupt for SIZE error event 0 Interrupt is disabled 1 Interrupt is enabled
3	NOMAP	Mask interrupt for NOMAP error event 0 Interrupt is disabled 1 Interrupt is enabled
4–15	—	Reserved, should be cleared.

21.2.4 Peripheral Interface

The CP uses the peripheral bus to communicate with all of its peripherals. Each FCC and each SCC has separate receive and transmit FIFOs. The FCC FIFOs are 192 bytes. The SCC FIFOs are 32 bytes. The SMCs, SPI, and I^2C are all double-buffered, creating effective FIFO sizes of two characters.

Table 21-5 shows the order in which the CP handles requests from peripherals from highest to lowest priority.

NOTE

Elevation to emergency status (priority 4) is determined on a per peripheral basis and may depend on a peripheral's mode of operation. Emergency prioritization among peripherals maintains relative normal prioritization. For example, simultaneous emergency requests from FCC1 transmit and SCC1 transmit would be handled in the same order as normal requests (FCC1 transmit).

Table 21-5. Peripheral Prioritization

Priority	Request
1	Reset in the CPCR or $\overline{\text{SRESET}}$
2	SDMA bus error
3	Commands issued to the CPCR
4	Emergency (from FCCs and SCCs)
5	USB
6	FCC1 receive
7	FCC1 transmit
8	FCC2 receive
9	FCC2 transmit
10	SCC1 receive
11	SCC1 transmit
12	SCC3 receive
13	SCC3 transmit
14	SCC4 receive
15	SCC4 transmit
16	Reserved
17	SMC1 receive
18	SMC1 transmit
19	SMC2 receive
20	SMC2 transmit
21	SPI receive
22	SPI transmit
23	I^2C receive
24	I^2C transmit
25	RISC timer table
26	Reserved

21.2.5 Execution from RAM

The CP has an option to execute microcode from a portion of user RAM located in the instruction RAM (IRAM). In this mode, the CP fetches instructions from both the IRAM and its own private ROM. This mode allows Freescale to add new protocols or enhancements to the MPC8555E in the form of RAM microcode packages. If preferred, the user can obtain binary microcode from Freescale and load it into the IRAM.

21.2.6 RISC Controller Configuration Register (RCCR)

The RISC controller configuration register (RCCR), shown in Figure 21-6, configures the CP to run microcode from ROM or instruction RAM and controls the CP's internal timer. The RCCR register is cleared at reset.

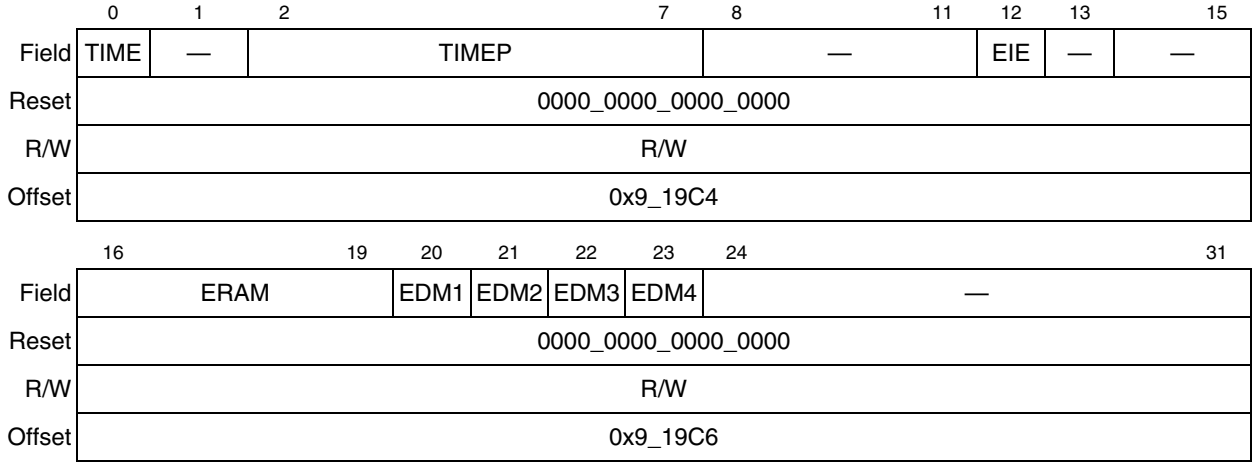


Figure 21-6. RISC Controller Configuration Register (RCCR)

RCCR bit fields are described in Table 21-6.

Table 21-6. RISC Controller Configuration Register Field Descriptions

Bits	Name	Description
0	TIME	Timer enable. Enables the CP internal timer that generates a tick to the CP based on the value programmed into the TIMEP field. TIME can be modified at any time to start or stop the scanning of the RISC timer tables.
1	—	Reserved, should be cleared.
2–7	TIMEP	Timer period controls the CP timer tick. The RISC timer tables are scanned on each timer tick and the input to the timer tick generator is the general system clock (133/166 MHz) divided by 1024. The formula is (TIMEP + 1) × 1024 = (general system clock period). Thus, a value of 0 stored in these bits gives a timer tick of 1 × (1024) = 1024 general system clocks and a value of 63 (decimal) gives a timer tick of 64 × (1024) = 65,536 general system clocks.
8–11	—	Reserved, should be cleared.
12	EIE	External interrupt enable. When EIE is set, DREQ1 acts as an external interrupt to the CP. Configure as instructed in the download process of a Freescale-supplied RAM microcode package. 0 DREQ1 cannot interrupt the CP. 1 DREQ1 will interrupt the CP.
13–15	—	Reserved, should be cleared.
16–19	ERAM	Enable RAM microcode. Configure as instructed in the download process of a Freescale-supplied RAM microcode package. Otherwise, it should not be used. 0000 Disable microcode program execution from the internal RAM. 0100 Microcode is executed from the instruction RAM. Other combinations of these bits are not valid and must not be used.

Table 21-6. RISC Controller Configuration Register Field Descriptions (continued)

Bits	Name	Description
20–23	EDM*x*	Edge detect mode. DREQ*x* asserts as follows: 0 Low-to-high change 1 High-to-low change
24–31	—	Reserved, should be cleared.

21.2.7 RISC Time-Stamp Control Register (RTSCR)

The RISC time-stamp control register (RTSCR), shown in Figure 21-7, configures the RISC time-stamp register (RTSR). The time-stamp timer is used by the ATM and the HDLC controllers. For application examples, see Section 41.5.3, "ABR Flow Control Setup," and Section 38.6, "HDLC Mode Register (FPSMR)."

	0 — 4	5	6 — 15
Field	—	RTE	RTPS (Timer Prescale)
Reset	0000_0000_0000_0000		
R/W	R/W		
Offset	0x9_19DC		

Figure 21-7. RISC Time-Stamp Control Register (RTSCR)

Table 21-7 describes RTSCR fields.

Table 21-7. RTSCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	RTE	Time stamp enable 0 Disable time-stamp timer 1 Enable time-stamp timer
6–15	RTPS	Time-stamp timer pre-scale. Must be programmed to generate a 1-µs period input clock to the time-stamp timer. Time-stamp frequency = (CPM frequency)/(RTPS + 2)

21.2.8 RISC Time-Stamp Register (RTSR)

The RISC time-stamp register (RTSR), shown in Figure 21-8, contains the time stamp.

	0	15
Field	Time Stamp	
Reset	—	
R/W	R	
Offset	0x9_19E0	

	16	31
Field	Time Stamp	
Reset	—	
R/W	R	
Offset	0x9_19E2	

Figure 21-8. RISC Time-Stamp Register (RTSR)

After reset, setting RTSCR[RTE] causes the time stamp to start counting microseconds from zero.

21.2.9 RISC Microcode Revision Number

The CP writes a revision number stored in its ROM to a dual-port RAM location called REV_NUM that resides in the miscellaneous parameter RAM. The other locations are reserved for future use. Table 21-8 describes the RISC microcode revision number.

Table 21-8. RISC Microcode Revision Number

Address	Name	Width	Description
RAM Base + 0x8AF0	REV_NUM	Hword	Microcode revision number. 0x00E8
RAM Base + 0x8AF2	—	Hword	Reserved

21.3 Command Set

The core issues commands to the CP by writing to the CP command register (CPCR). The CPCR rarely needs to be accessed. For example, to terminate the transmission of an SCC's frame without waiting until the end, a STOP TX command must be issued through the CPCR.

21.3.1 CP Command Register (CPCR)

The core sets CPCR[FLG], shown in Figure 21-9, when it issues a command. The CP clears FLG after completing the command to indicate to the core that it is ready for the next command. Subsequent commands to the CPCR can be given only after FLG is clear. However, the software reset command, issued by setting RST, does not depend on the state of FLG, although the core should still set FLG when setting RST.

Table 21-9 describes CPCR fields.

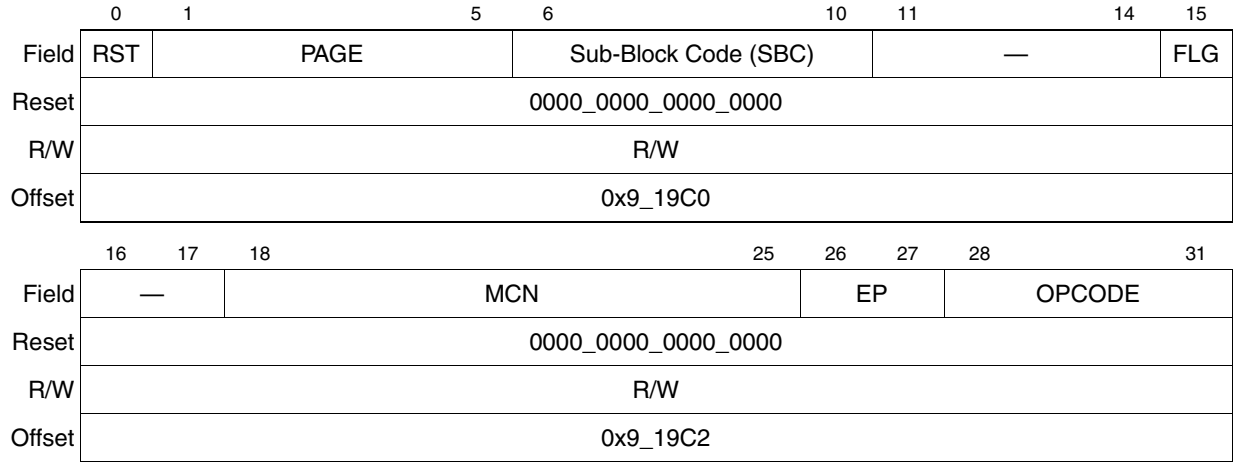


Figure 21-9. CP Command Register (CPCR)

Table 21-9. CP Command Register Field Descriptions

Bits	Name	Description
0	RST	Software reset command. Set by the core and cleared by the CP. When this command is executed, RST and FLG bit are cleared within two general system clocks. The CPM reset routine is approximately 60 clocks long, but the user can begin initialization of the CPM immediately after this command is issued. RST is useful when the core wants to reset the registers and parameters for all the channels (FCCs, SCCs, SPI, I^2C) as well as the CP and RISC timer tables. However, this command does not affect the serial interface (SI*x*) or parallel I/O registers.
1–5	PAGE	Indicates the parameter RAM page number associated with the sub-block being served. See the SBC description for page numbers.
6–10	SBC	Sub-block code. Set by the core to specify the sub-block on which the command is to operate. Set according to OPCODE (bits 28–31).
11–14	—	Reserved

Sub-Block	Code	Page	Sub-Block	Code	Page
FCC1[1]	01110: ATM transmit (OPCODE = 1010) 10000: all other commands	00100	SPI	01010	01001
FCC2[1]	01110: ATM transmit (OPCODE = 1010) 10001: all other commands	00101	I^2C	01011	01010
SCC1	00100	00000	Timer	01111	01010
SCC3	00110	00010	Reserved	10100	00111
SCC4	00111	00011	Reserved	10101	01000
SMC1	01000	00111	Reserved	10110	01001
SMC2	01001	01000	Reserved	10111	01010
RAND	01110	01010	USB	10011	01011

Table 21-9. CP Command Register Field Descriptions (continued)

Bits	Name	Description
15	FLG	Command semaphore flag. Set by the core and cleared by the CP 0 The CP is ready to receive a new command. 1 The CPCR contains a command that the CP is currently processing. The CP clears this bit at the end of command execution or after reset.
16–17	—	Reserved
18–25	MCN	In FCC protocols, this field contains the protocol code as follows: 0x00 HDLC 0x0A ATM 0x0C Ethernet 0x0F Transparent
26–27	EP	Endpoint: Logical pipe number (only in USB) 00 Endpoint 0 01 Endpoint 1 10 Endpoint 2 11 Endpoint 3
28–31	OPCODE	Operation code. Settings are listed in Table 21-10.

[1] Set according to OPCODE[28–31]. If OPCODE is 1010, SBC must be 01110. Refer to Table 21-10.

21.3.1.1 CP Commands

The CP command opcodes are shown in Table 21-10.

Table 21-10. CP Command Opcodes

Opcode	Channel								
	FCC	USB	SCC	SMC (UART/ Transparent)	SMC (GCI)	SPI	I^2C	Timer	Special
0000	INIT RX AND TX PARAMS	—	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	—	—
0001	INIT RX PARAMS	—	INIT RX PARAMS	INIT RX PARAMS	—	INIT RX PARAMS	INIT RX PARAMS	—	—
0010	INIT TX PARAMS	—	INIT TX PARAMS	INIT TX PARAMS	—	INIT TX PARAMS	INIT TX PARAMS	—	—
0011	ENTER HUNT MODE	—	ENTER HUNT MODE	ENTER HUNT MODE	—	—	—	—	—
0100	STOP TX	—	STOP TX	STOP TX	—	—	—	—	—
0101	GRACEFUL STOP TX	—	GRACEFUL STOP TX	—	—	—	—	—	—
0110	RESTART TX	—	RESTART TX	RESTART TX	—	—	—	—	—
0111	—	—	CLOSE RX BD	CLOSE RX BD	—	CLOSE RX BD	CLOSE RX BD	—	—

Table 21-10. CP Command Opcodes (continued)

Opcode	Channel								
	FCC	USB	SCC	SMC (UART/ Transparent)	SMC (GCI)	SPI	I^2C	Timer	Special
1000	SET GROUP ADDRESS	—	SET GROUP ADDRESS	—	—	—	—	SET TIMER	—
1001	—	—	—	—	GCI TIMEOUT	—	—	—	—
1010	ATM TRANSMIT COMMAND[1]	USB STOP TX ENDPOINT	RESET BCS	—	GCI ABORT REQUEST	—	—	—	—
1011	—	USB RESTART TX ENDPOINT	—	—	—	—	—	—	—
1100	—	—	—	—	—	—	—	—	RANDOM NUMBER
11XX	Undefined. Reserved for use by Freescale-supplied RAM microcodes.								

[1] See SBC for FCC1 and FCC2, Table 21-9.

NOTE

If a reserved command is issued, the CPM enters an unknown state that requires an external reset to recover.

The commands in Table 21-10 are described in Table 21-11.

Table 21-11. Command Descriptions

Command	Description
INIT TX AND RX PARAMS	Initialize transmit and receive parameters. Initializes the transmit and receive parameters in the parameter RAM to the values that they had after the last reset of the CP. This command is especially useful when switching protocols on a given peripheral controller.
INIT RX PARAMS	Initialize receive parameters. Initializes the receive parameters of the peripheral controller.
INIT TX PARAMS	Initialize transmit parameters. Initializes the transmit parameters of the peripheral controller.
ENTER HUNT MODE	Enter hunt mode. Causes the receiver to stop receiving and begin looking for a new frame. The exact operation of this command may vary depending on the protocol used.
STOP TX	Stop transmission. Aborts the transmission from this channel as soon as the transmit FIFO has been emptied. It should be used in cases where transmission needs to be stopped as quickly as possible. Transmission proceeds when the RESTART command is issued.
GRACEFUL STOP TX	Graceful stop transmission. Stops the transmission from this channel as soon as the current frame has been fully transmitted from the transmit FIFO. Transmission proceeds when the RESTART command is issued and the R-bit is set in the next TxBD.
RESTART TX	Restart transmission. Once the STOP TX command has been issued, this command is used to restart transmission at the current BD.

Table 21-11. Command Descriptions (continued)

Command	Description
USB STOP TX ENDPOINT	See Section 35.7, “USB CP Commands.”
USB RESTART TX ENDPOINT	See Section 35.7, “USB CP Commands.”
CLOSE RXBD	Close RxBD. Causes the receiver to close the current RxBD, making the receive buffer immediately available for manipulation by the user. Reception continues using the next available BD. Can be used to access the buffer without waiting until the buffer is completely filled by the SCC.
SET TIMER	Set timer. Activates, deactivates, or reconfigures one of the 16 timers in the RISC timer table.
SET GROUP ADDRESS	Set group address. Sets a bit in the hash table for the Ethernet logical group address recognition function.
GCI ABORT REQUEST	GCI abort request. The GCI receiver sends an abort request on the E bit.
GCI TIMEOUT	GCI time-out. The GCI performs the timeout function.
RESET BCS	Reset block check sequence. Used in BISYNC mode to reset the block check sequence calculation.
ATM TRANSMIT	See Section 41.14, “ATM Transmit Command.”
RANDOM NUMBER	Generate a random number and put it in dual-port RAM; see RAND in Table 21-13.

NOTE

The CPM accesses BDs by initiating a DMA cycle on either the system or local bus. If BDs are located in DPRAM, the CPM initiates a cycle on the system bus because DPRAM is a slave device on the system bus. Therefore, a system design should not plan to access the system bus simultaneously with DPRAM BD fetches.

21.3.2 Command Register Example

To perform a complete reset of the CP, the value 0x8001_0000 should be written to the CPCR. Following this command, the CPCR returns the value 0x0000_0000 after two clocks.

21.3.3 Command Execution Latency

The worst-case command execution latency is 200 clocks and the typical command execution latency is about 40 clocks.

21.4 Internal RAM

The CPM has 20 Kbytes of static RAM. This RAM is split into two blocks.

- 4 Kbytes of instruction RAM to store a microcode package of up to 1K instructions
- 16 Kbytes of dual-port data RAM to store CPM-RISC parameter RAM and data structures

Figure 21-10 is a block diagram of the internal RAM.

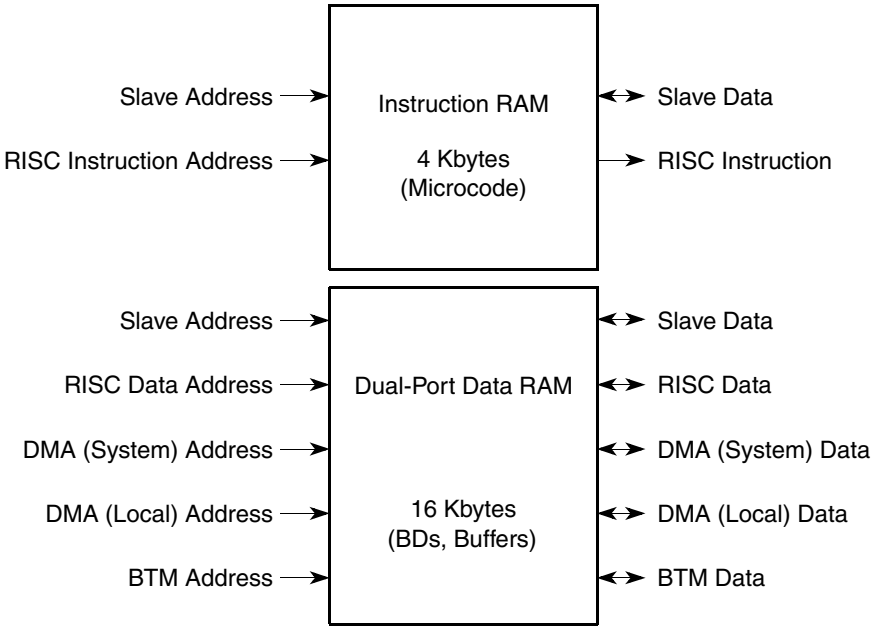


Figure 21-10. Internal RAM Block Diagram

The internal instruction RAM can be accessed by the following:

- CP instruction fetcher (in case of microcode from RAM)
- Other masters connected through the ECM (including the processor core)

The internal dual-port data RAM can be accessed by the following:

- CP load/store unit
- CP block transfer module (BTM)
- Other masters connected through the ECM (including the processor core)
- SDMA system bus
- SDMA local bus

Figure 21-11 shows the memory map of the internal instruction RAM.

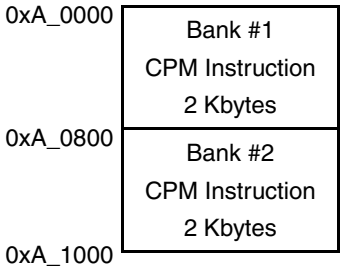


Figure 21-11. Internal Instruction RAM Memory Map

Figure 21-12 shows a memory map of the internal dual-port data RAM.

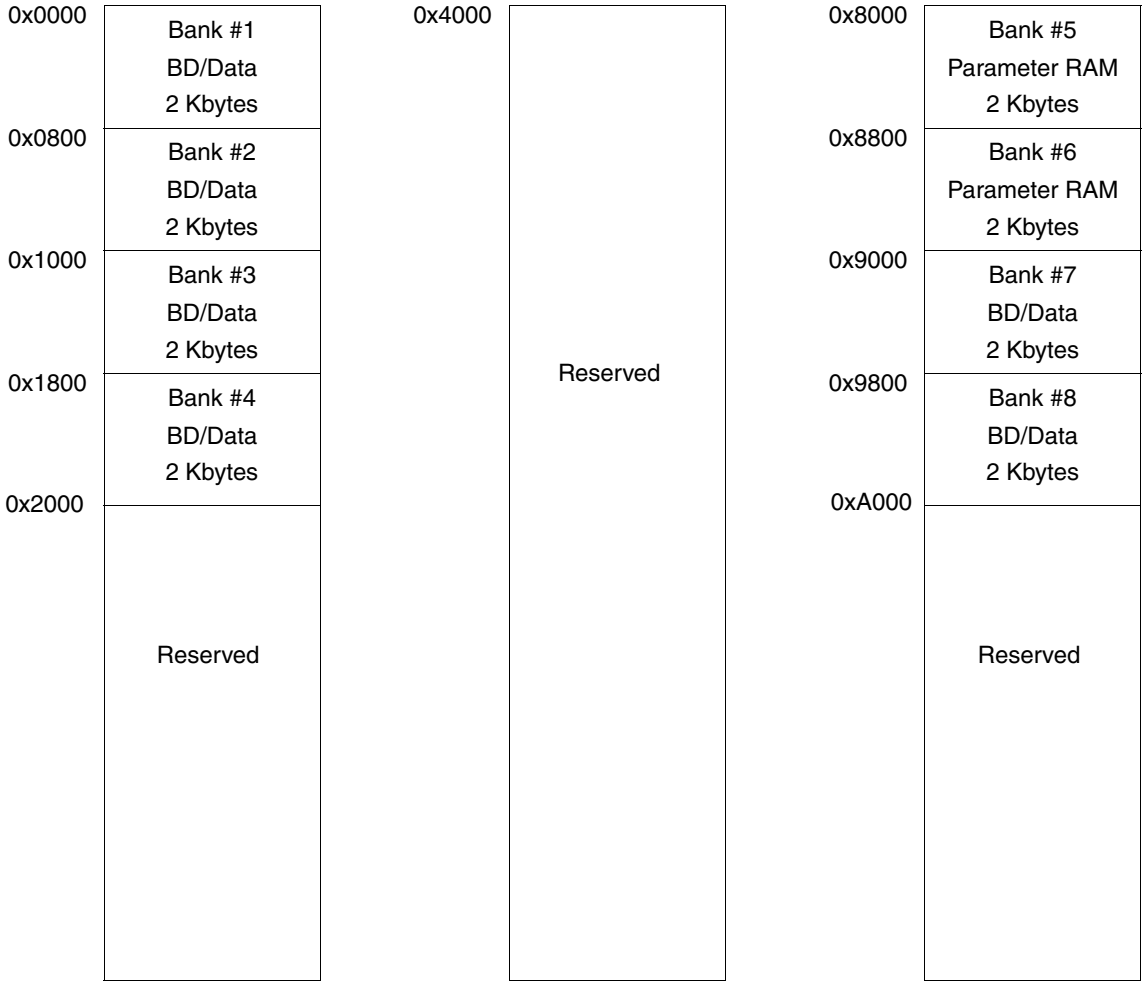


Figure 21-12. Internal Dual-Port Data RAM Memory Map

The dual-port data RAM data bus is 64 bits wide. The RAM is used for the following tasks:

- To store parameters associated with the FCCs, SCCs, SMCs, SPI, and I^2C in the parameter RAM
- To store buffer descriptors (BDs)
- To hold data buffers (optional because data can also be stored in external memory)
- To temporarily store FCC data moving to/from an FCC FIFO (using the BTM) from/to external memory (using SDMA)
- As an additional scratch-pad RAM space for user software

The dual-port data RAM is designed to serve multiple requests at the same cycle, as long as they are not in the same bank.

Only the parameters in the parameter RAM and the microcode RAM option require fixed addresses to be used. The BDs, buffer data, and scratchpad RAM can be located in the internal system RAM or in any

unused parameter RAM, for example, in the area made available when a peripheral controller or sub-block is not being used.

Microcode can be executed from the instruction RAM (separate from the data RAM).

21.4.1 Buffer Descriptors (BDs)

The peripheral controllers (FCCs, SCCs, SMCs, SPI, and I^2C) always use BDs for controlling buffers and their BD formats, shown in Table 21-12, are all the same.

Table 21-12. Buffer Descriptor Format

Address	Descriptor
Offset + 0	Status and control
Offset + 2	Data length
Offset + 4	High-order buffer pointer
Offset + 6	Low-order buffer pointer

21.4.2 Parameter RAM

The CPM maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the FCCs, SCCs, SMCs, SPI, and I^2C channels. An overview of the parameter RAM structure is shown in Table 21-13.

The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM. For example, the Ethernet parameter RAM is defined differently in some locations from the HDLC-specific parameter RAM.

Table 21-13. Parameter RAM

Page	Address [1]	Peripheral	Size (Bytes)
1	0x8000	SCC1	256
2	0x8100	Reserved	256
3	0x8200	SCC3	256
4	0x8300	SCC4	256
5	0x8400	FCC1	256
6	0x8500	FCC2	256
7	0x8600	Reserved	256

Table 21-13. Parameter RAM (continued)

Page	Address [1]	Peripheral	Size (Bytes)
8	0x8700	Reserved	128
	0x8780	Reserved	124
	0x87FC	SMC1_BASE	2
	0x87FE	Reserved	2
9	0x8800	Reserved	128
	0x8880	Reserved	124
	0x88FC	SMC2_BASE	2
	0x88FE	Reserved	2
10	0x8900	Reserved	252
	0x89FC	SPI_BASE	2
	0x89FE	Reserved	2
11	0x8A00	Reserved	224
	0x8AE0	RISC Timers	16
	0x8AF0	REV_NUM[2]	2
	0x8AF2	Reserved	2
	0x8AF4	Reserved	4
	0x8AF8	RAND	4
	0x8AFC	I^2C_BASE	2
	0x8AFE	Reserved	2
12	0x8B00	USB	256
13-16	0x8C00	Reserved	1024

1 Offset from RAM_BASE.
2 Refer to Table 21-8.

21.5 RISC Timer Tables

The CP can control up to 16 software timers that are separate from the 4 general-purpose timers and the BRGs in the CPM. These timers are best used in protocols that do not require extreme precision, but in which it is preferable to free the core from scanning the software's timer tables. These timers are clocked from an internal timer that only the CP uses. The following is a list of the important features of the RISC timer tables.

- Up to 16 timers supported
- Two timer modes: one-shot and restart
- Maskable interrupt on timer expiration
- Programmable timer resolution as fine as 3.85 µs at 266 MHz (3.09 µs at 333 MHz)

- Maximum timeout period of 15.9 seconds at 266 MHz (12.8 seconds at 333 MHz)
- Continuously updated reference counter

All operations on the RISC timer tables are based on a fundamental tick of the CP's internal timer that is programmed in the RCCR. The tick is a multiple of 1024 general system clocks; see Section 21.2.6, "RISC Controller Configuration Register (RCCR)."

The RISC timer tables have the lowest priority of all CP operations. Therefore, if the CP is so busy with other tasks that it does not have time to service the timer during a tick interval, one or more timers may not be updated accurately. This behavior can be used to estimate the worst-case loading of the CP; see Section 21.5.10, "Using the RISC Timers to Track CP Loading."

The timer table is configured using the RCCR, the timer table parameter RAM, and the RISC controller timer event/mask registers (RTER/RTMR), and by issuing SET TIMER to the CPCR.

21.5.1 RISC Timer Table Parameter RAM

Two areas of dual-port RAM, shown in Figure 21-13, are used for the RISC timer tables:

- The RISC timer table parameter RAM
- The RISC timer table entries

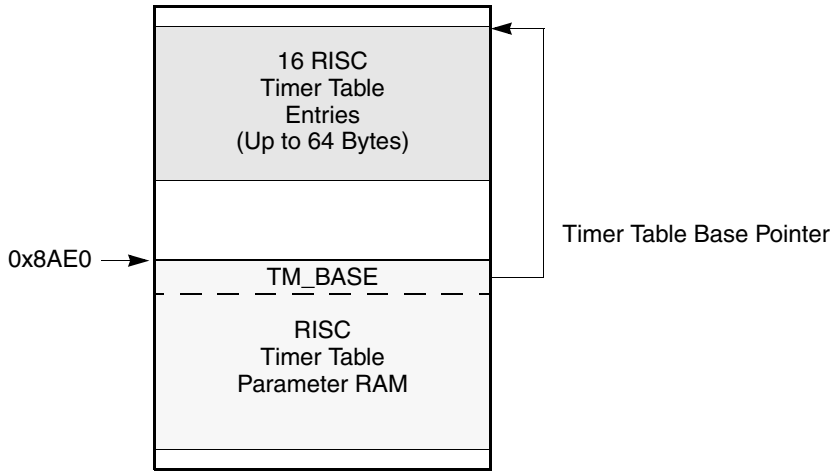


Figure 21-13. RISC Timer Table RAM Usage

The RISC timer table parameter RAM area begins at the RISC timer base address and is used for the general timer parameters; see Table 21-14.

Table 21-14. RISC Timer Table Parameter RAM

Offset[1]	Name	Description
0x00	TM_BASE	RISC timer table base address. The actual timers are a small block of memory in the dual-port RAM. TM_BASE is the offset from the beginning of the dual-port RAM where that block resides. Four bytes must be reserved at the TM_BASE for each timer used (64 bytes if all 16 timers are used). If fewer than 16 timers are used, timers should be allocated in ascending order to save space. For example, only 8 bytes are required if two timers are needed and RISC timers 0 and 1 are enabled. TM_BASE should be word-aligned.

Table 21-14. RISC Timer Table Parameter RAM (continued)

0x02	TM_PTR	RISC timer table pointer. This value is used exclusively by the CP to point to the next timer accessed in the timer table. It should not be modified by the user.
0x04	R_TMR	RISC timer mode register. This value is used exclusively by the CP to store the mode of the timer—one-shot (bit is 0) or restart (bit is 1). R_TMR should not be modified by the user. The SET TIMER command should be used instead.
0x06	R_TMV	RISC timer valid register. Used exclusively by the CP to determine if a timer is currently enabled. If the corresponding timer is enabled, a bit is 1. R_TMV should not be modified by the user. The SET TIMER command should be used instead.
0x08	TM_CMD	RISC timer command register. Used as a parameter location when the SET TIMER command is issued. The user should write this location before issuing the SET TIMER command. This register is defined in Section 21.5.2, “RISC Timer Command Register (TM_CMD).”
0x0C	TM_CNT	RISC timer internal count. A tick counter that the CP updates after each tick. The update occurs after the CP complete scanning the timer table.All 16 timers are scanned every tick interval regardless of whether any of them is enabled.It is updated if the CP’s internal timer is enabled, regardless of whether any of the 16 timers are enabled and it can be used to track the number of ticks the CP receives and responds to.TM_CNT is updated only after the last timer (timer 15) has been serviced. If the CP is so busy with other tasks that it does not have time to service all the timers during a tick interval, and timer 15 has not been serviced, then TM_CNT would not be updated in that tick interval.

[1] Offset from timer base address (0x8AE0).

21.5.2 RISC Timer Command Register (TM_CMD)

Figure 21-14 shows the RISC timer command register (TM_CMD).

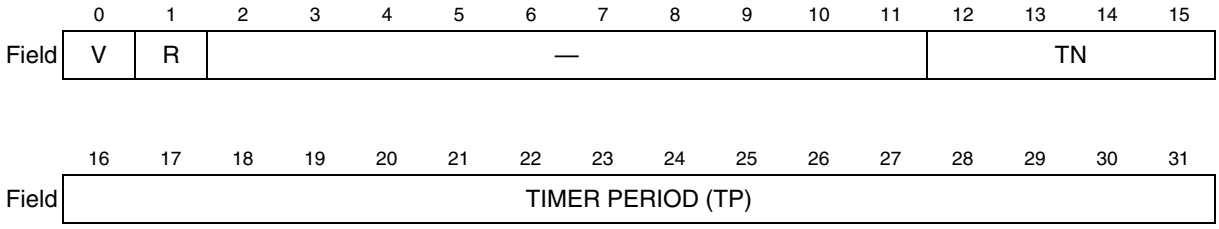


Figure 21-14. RISC Timer Command Register (TM_CMD)

TM_CMD fields are described in Table 21-15.

Table 21-15. TM_CMD Field Descriptions

Bits	Name	Description
0	**V**	Valid. This bit should be set to enable the timer and cleared to disable it.
1	**R**	Restart. Should be set for an automatic restart or cleared for a one-shot operation of the timer.
2–11	—	Reserved. These bits should be written with zeros.
12–15	**TN**	Timer number. A value from 0–15 signifying which timer to use—an offset into the timer table entries.
16–31	**TP**	Timer period. The 16-bit timeout value of the timer is zero-based. The minimum value is 1 and is programmed by writing 0x0000 to the timer period.The maximum value of the timer is 65,536 and is programmed by writing 0xFFFF.

21.5.3 RISC Timer Table Entries

The 16 timers are located in the block of memory following the TM_BASE location; each timer occupies 4 bytes. The first half word forms the initial value of the timer written during the execution of the SET TIMER command and the next half word is the current value of the timer that is decremented until it reaches zero. These locations should not be modified by the user. They are documented only as a debugging aid for user code. Use the SET TIMER command to initialize table values.

21.5.4 RISC Timer Event Register (RTER)/Mask Register (RTMR)

The RTER register is used to report events recognized by the 16 timers and to generate interrupts. RTER can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values.

The RISC timer mask register (RTMR) is used to enable interrupts that can be generated in the RTER. Setting an RTMR bit enables the corresponding interrupt in the RTER; clearing a bit masks the corresponding interrupt. An interrupt is generated only if the RISC timer table bit is set in the CPM interrupt mask register. See Section 22.5.1.4, “CPM Interrupt Mask Registers (SIMR_H, SIMR_L).”

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMR 15	TMR 14	TMR 13	TMR 12	TMR 11	TMR 10	TMR 9	TMR 8	TMR 7	TMR 6	TMR 5	TMR 4	TMR 3	TMR 2	TMR 1	TMR 0
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_19D6 (RTER)/0x9_19DA (RTMR)															

Figure 21-15. RISC Timer Event Register (RTER)/Mask Register (RTMR)

21.5.5 SET TIMER Command

The SET TIMER command is used to enable, disable, and configure the 16 timers in the RISC timer table and is issued to the CPCR. This means the value 0x29E1_0008 should be written to CPCR. However, before writing this value, the user should program the TM_CMD fields. See Section 21.5.2, “RISC Timer Command Register (TM_CMD).”

21.5.6 RISC Timer Initialization Sequence

The following sequence initializes the RISC timers:

1. Configure RCCR to determine the preferred tick interval for the entire timer table. The TIME bit is normally set at this time but can be set later if all RISC timers need to be synchronized.
2. Determine the maximum number of timers to be located in the timer table. Configure the TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with 4 × n bytes available, where n is the number of timers. If n is less than 16, use timer 0 through timer n–1 to save space.
3. Clear the TM_CNT field in the RISC timer table parameter RAM to show how many ticks elapsed since the RISC internal timer was enabled. This step is optional.

4. Clear RTER, if it is not already cleared. Write ones to clear this register.
5. Configure RTMR to enable the timers that should generate interrupts. Ones enable interrupts.
6. Set the RISC timer table bit in the CPM interrupt mask register (SIMR_L[RTT]) to generate interrupts to the system. The CPM interrupt controller may require other initialization not mentioned here.
7. Configure the TM_CMD field of the RISC timer table parameter RAM. At this point, determine whether a timer is to be enabled or disabled, one-shot or restart, and what its timeout period should be. If the timer is being disabled, the parameters (other than the timer number) are ignored.
8. Issue the SET TIMER command by writing 0x29E1_0008 to the CPCR.
9. Repeat the preceding two steps for each timer to be enabled or disabled.

21.5.7 RISC Timer Initialization Example

The following sequence initializes RISC timer 0 to generate an interrupt approximately every second using a 133-MHz general system clock:

1. Write 111111 to RCCR[TIMEP] to generate the slowest clock. This value generates a tick every 65,536 clocks, which is every 485 µs at 133 MHz.
2. Configure the TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with 4 bytes available. Assuming the beginning of dual-port RAM is available, write 0x0000 to TM_BASE.
3. (Optional) Write 0x0000 to the TM_CNT field in the RISC timer table parameter RAM to see how many ticks elapsed since the RISC internal timer was enabled.
4. Write 0xFFFF to the RTER to clear any previous events.
5. Write 0x0001 to the RTMR to enable RISC timer 0 to generate an interrupt.
6. Write 0x0002_0000 to the CPM interrupt mask register (SIMR_L) to allow the RISC timers to generate a system interrupt. Initialize the CPM interrupt configuration register.
7. Write 0xC000_080D to the TM_CMD field of the RISC timer table parameter RAM. This enables RISC timer 0 to timeout after 2,061(decimal) ticks of the timer. The timer automatically restarts after it times out.
8. Write 0x29E1_0008 to the CPCR to issue the SET TIMER command.
9. Set RCCR[TIME] to enable the RISC timer to begin operation.

21.5.8 RISC Timer Interrupt Handling

The following sequence describes what normally would occur within an interrupt handler for the RISC timer tables:

1. Once an interrupt occurs, read RTER to see which timers have caused interrupts. The RISC timer event bits are usually cleared by this time.
2. Issue additional SET TIMER commands at this time or later, as preferred. Nothing needs to be done if the timer is being automatically restarted for a repetitive interrupt.

3. Clear the RTT bit in the CPM interrupt pending register (SIPNR_L).
4. Execute the RTE instruction.

21.5.9 RISC Timer Table Scan Algorithm

The CP scans the timer table once every tick. It handles each of the 16 timers at its turn and checks for other requests with higher priority to service before handling the next one. For each valid timer in the table, the CP decrements the count and checks for a timeout. If none occurs, the CP moves to the next timer. If a timeout occurs, the CP sets the corresponding event bit in RTER. Then the CP checks to see if the timer is to be restarted and if it is, the CP leaves the timer's valid bit set in the R_TMV location and resets the current count to the initial count. Otherwise, it clears R_TMV. Once the timer table scanning has completed, the CP updates the TM_CNT value in the RISC timer table parameter RAM and stops working on the timer tables until the next tick.

If a SET TIMER command is issued, the CP makes the appropriate modifications to the timer table and parameter RAM, but does not scan the timer table until the next tick of the internal timer. It is important to use the SET TIMER command to properly synchronize timer table modifications to the execution of the CP.

21.5.10 Using the RISC Timers to Track CP Loading

The RISC timers can be used to track CP loading. The following sequence provides a way to use the 16 RISC timers to determine if the CP ever exceeds the 96% utilization level during any tick interval. Removing the timers adds a 4% margin to the CP utilization level, but the aggressive user can use this technique to push CP performance to its limit. The user should use the standard initialization sequence and incorporate the following differences:

1. Program the tick of the RISC timers to be every $1024 \times 16 = 16{,}384$ system clocks.
2. Disable RISC timer interrupts, if preferred.
3. Using the SET TIMER command, initialize all 16 RISC timers to have a timer period of 0xFFFF, which is equal to 65,536.
4. Program one of the four general-purpose timers to increment once every tick. The general-purpose timer should be free-running and should have a timeout of 65,536.
5. After a few hours of operation, compare the general-purpose timer to the current count of RISC timer 15. If it is more than two ticks different from the general-purpose timer, the CP has, during some tick interval, exceeded the 96% utilization level.

NOTE

General-purpose timers are up counters, but RISC timers are down counters. The user should take this into consideration when comparing timer counts.

Chapter 22
CPM Interrupt Controller

Key features of the CPM interrupt controller include the following:

- Communications processor module (CPM) interrupt sources (FCCs, SCCs, SMCs, timers, I^2C, SDMA, USB, and SPI)
- 16 external interrupt pins (port C)
- Programmable priority between SCCs and FCCs
- Two priority schemes for the SCCs: grouped, spread
- Programmable highest priority request
- Unique vector number for each interrupt source

22.1 Interrupt Configuration

Figure 22-1 shows the MPC8555E interrupt structure. The interrupt controller receives interrupts from internal sources from the CPM and from external pins (port C parallel I/O pins). The CPM interrupt controller combines all of these interrupt inputs into a single interrupt signal to the programmable interrupt controller (PIC), which treats it as an internal interrupt.

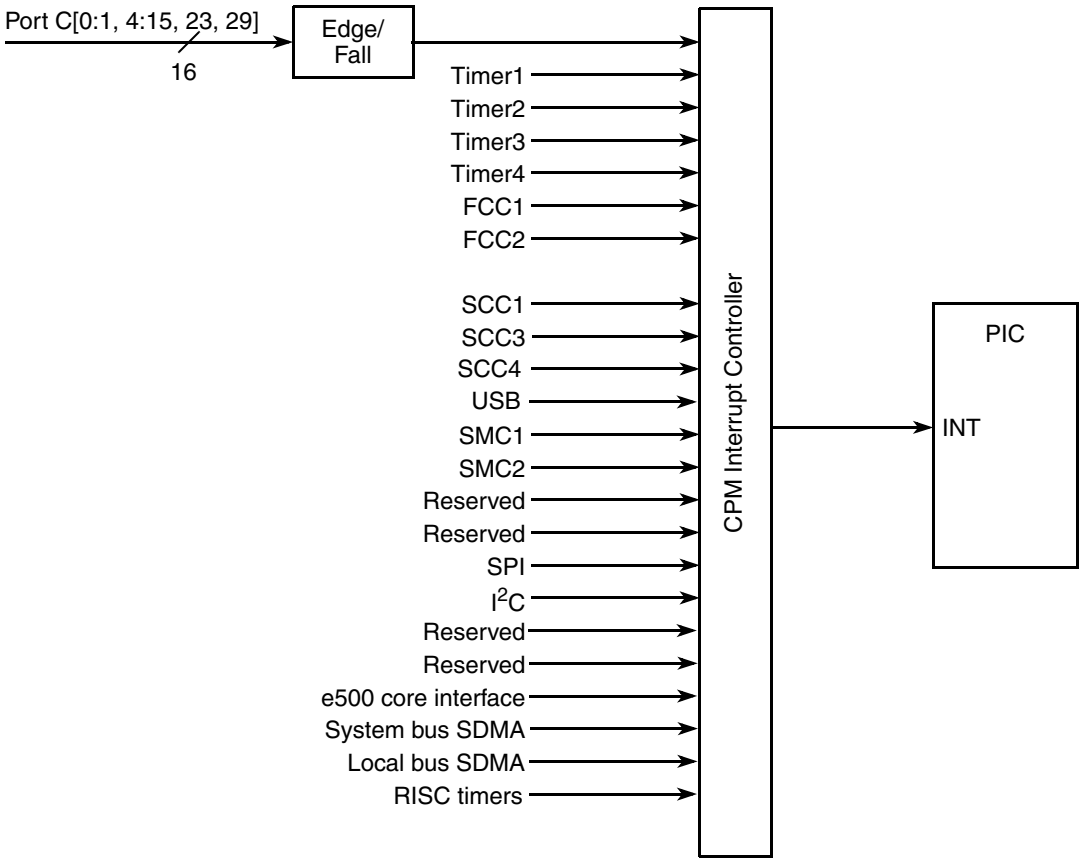


Figure 22-1. MPC8555E CPM Interrupt Structure

The CPM interrupt controller allows masking of each interrupt source. Multiple events within a CPM sub-block event are also maskable.

All interrupt sources are prioritized and bits are set in the interrupt pending register (SIPNR). On the MPC8555E, the prioritization of the interrupt sources is flexible in the following two aspects:

- The relative priority of the FCCs and SCCs can be modified.
- One interrupt source can be assigned the highest priority.

When an unmasked interrupt source is pending in the SIPNR, the interrupt controller sends an interrupt request to the PIC.

The CPM interrupt vector register (SIVEC) is updated with a 6-bit vector corresponding to the requesting sub-block with the highest current priority.

22.2 CPM Interrupt Source Priorities

The CPM interrupt controller has 34 interrupt sources that assert one interrupt request to the PIC. Table 22-1 shows the prioritization of the interrupt sources. As described in the following 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.

Note that the group and spread options, shown with YCC entries in Table 22-1, are described in Section 22.2.1, “SCC and FCC Relative Priority.”

Table 22-1. Interrupt Source Priority Levels

Priority Level	Interrupt Source Description	Multiple Events
1	Highest	—
2	XCC1	Yes
3	XCC2	Yes
4	XCC3	Yes
5	XCC4	Yes
6	XCC5	Yes
7	XCC6	Yes
8	XCC7	Yes
9	XCC8	Yes
10	YCC1 (grouped)	Yes
11	YCC2 (grouped)	Yes
12	YCC3 (grouped)	Yes
13	YCC4 (grouped)	Yes
14	YCC5 (grouped)	Yes
15	YCC6 (grouped)	Yes
16	YCC7 (grouped)	Yes
17	YCC8 (grouped)	Yes
18	Parallel I/O–PC29	Yes
19	Timer 1	Yes
20	Parallel I/O–PC23	Yes
21	YCC1 (spread)	Yes
22	Parallel I/O–PC15	Yes
23	SDMA bus error	Yes
24	USB	Yes
25	YCC2 (spread)	Yes
26	Parallel I/O–PC14	No
27	Parallel I/O–PC13	No
28	Reserved	—
29	Timer 2	Yes
30	Parallel I/O–PC12	No
31	YCC3 (spread)	Yes

Table 22-1. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
32	RISC timer table	Yes
33	I^2C	Yes
34	YCC4 (spread)	Yes
35	Parallel I/O–PC11	No
36	Parallel I/O–PC10	No
37	Reserved	—
38	Timer 3	Yes
39	YCC5 (spread)	Yes
40	Parallel I/O–PC9	No
41	Parallel I/O–PC8	No
42	Parallel I/O–PC7	No
43	Timer 4	Yes
44	YCC6 (spread)	Yes
45	Parallel I/O–PC6	No
46	Reserved	—
47	SPI	Yes
48	Parallel I/O–PC5	No
49	Parallel I/O–PC4	No
50	SMC1	Yes
51	YCC7 (spread)	Yes
52	SMC2	Yes
53	Parallel I/O–PC1	No
54	Parallel I/O–PC0	No
55	YCC8 (spread)	Yes
56	Reserved	—

Notice the lack of SDMA interrupt sources, which are reported through each individual FCC, SCC, SMC, SPI, or I^2C channel. The only true SDMA interrupt source is the SDMA channel bus error entry that is reported when a bus error occurs during an SDMA access. There are two ways to add flexibility to the table of CPM interrupt priorities—the FCC and SCC relative priority option, described in Section 22.2.1, “SCC and FCC Relative Priority,” and the highest priority option, described in Section 22.2.2, “Highest Priority Interrupt.”

22.2.1 SCC and FCC Relative Priority

The relative priority between the three SCCs and two FCCs is programmable and can be changed dynamically. In Table 22-1 there is no entry for SCC1–SCC4, FCC1–FCC2, but rather there are entries for XCC1–XCC8 and YCC1–YCC8. Each SCC can be mapped to any YCC location and each FCC can be mapped to any XCC location. The SCC and FCC priorities are programmed in the CPM interrupt priority registers (SCPRR_H and SCPRR_L) and can be changed dynamically to implement a rotating priority.

In addition, grouping of YCC entry locations can occur in one of the following two ways:

- Group. In the group scheme, all SCCs are grouped together at the top of the priority table, ahead of most other CPM interrupt sources. This scheme is ideal for applications where all SCCs and FCCs 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 in the SICR but cannot be changed dynamically.

22.2.2 Highest Priority Interrupt

In addition to the FCC/SCC relative priority option, SICR[HP] can be used to specify one interrupt source as having highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but is serviced before any other interrupt in the table.

If the highest priority feature is not used, select the interrupt request with the highest priority. SICR[HP] can be updated dynamically to allow the user to change a normally low priority source into a high priority-source for a certain period.

22.3 Masking Interrupt Sources

By programming the CPM interrupt mask registers, SIMR_H and SIMR_L, the user can mask interrupt requests to the core. Each SIMR bit corresponds to an interrupt source. To enable an interrupt, set the corresponding SIMR bit. When a masked interrupt source has a pending interrupt request, the corresponding SIPNR 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 block. Table 22-1 shows which interrupt sources have multiple interrupting events. Figure 22-2 shows an example of how the masking occurs, using an SCC as an example.

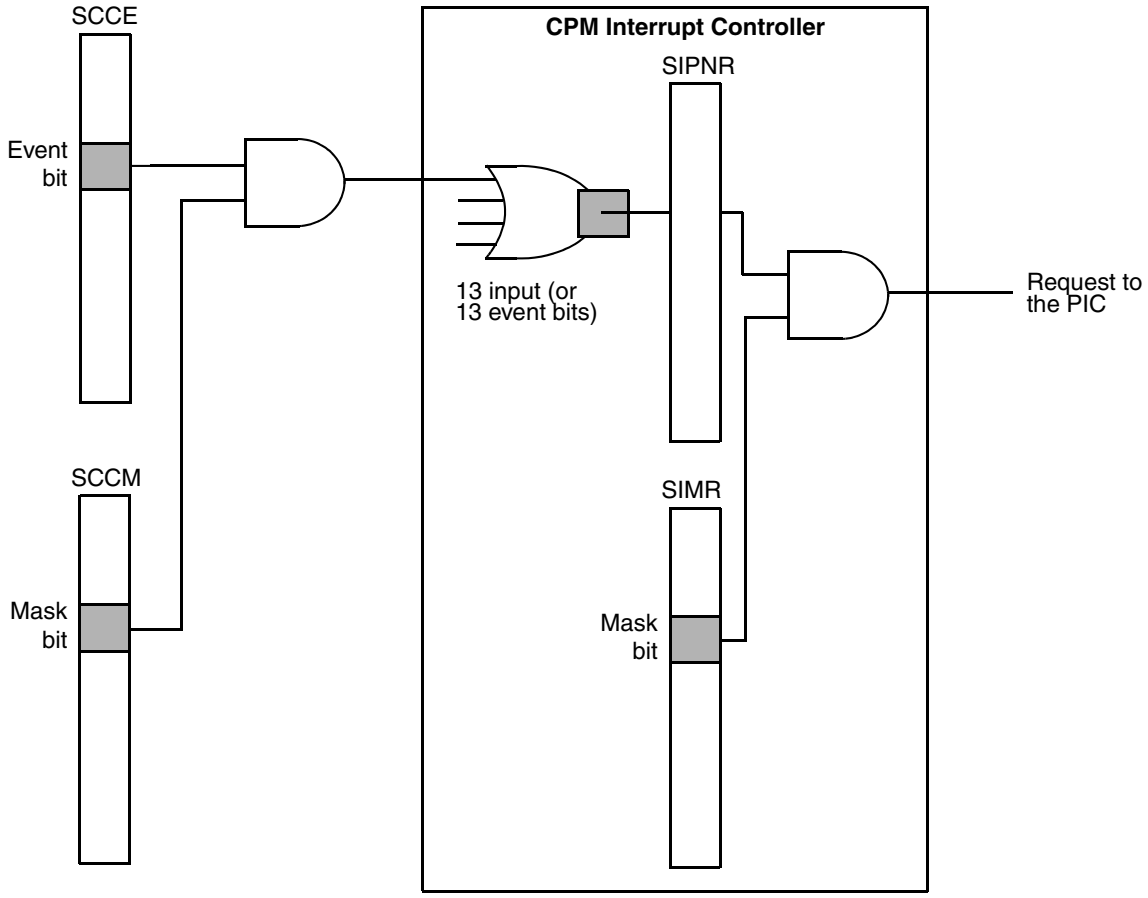


Figure 22-2. Interrupt Request Masking

22.4 CPM Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority. The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by reading SIVEC. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt. Table 22-2 lists encodings for the 6 low-order bits of the interrupt vector.

Table 22-2. Encoding the Interrupt Vector

Interrupt Number	Interrupt Source Description	Interrupt Vector
0	Error (no interrupt)	0b00_0000
1	I^2C	0b00_0001
2	SPI	0b00_0010
3	RISC timers	0b00_0011
4	SMC1	0b00_0100
5	SMC2	0b00_0101
6	Reserved	0b00_0110

Table 22-2. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
7	Reserved	0b00_0111
8	Reserved	0b00_1000
9	Reserved	0b00_1001
10	Reserved	0b00_1010
11	USB	0b00_1011
12	Timer1	0b00_1100
13	Timer2	0b00_1101
14	Timer3	0b00_1110
15	Timer4	0b00_1111
16–31	Reserved	0b01_0000–01_1111
32	FCC1	0b10_0000
33	FCC2	0b10_0001
34	Reserved	0b10_0010
35	Reserved	0b10_0011
36	Reserved	0b10_0100
37	Reserved	0b10_0101
38	Reserved	0b10_0110
39	Reserved	0b10_0111
40	SCC1	0b10_1000
41	Reserved	0b10_1001
42	SCC3	0b10_1010
43	SCC4	0b10_1011
44	Reserved	0b10_1100
45	Core interface	0b10_1101
46	SDMA system	0b10_1110
47	SDMA local	0b10_1111
48	PC29	0b11_0000
49	PC23	0b11_0001
50	PC15	0b11_0010
51	PC14	0b11_0011
52	PC13	0b11_0100

Table 22-2. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
53	PC12	0b11_0101
54	PC11	0b11_0110
55	PC10	0b11_0111
56	PC9	0b11_1000
57	PC8	0b11_1001
58	PC7	0b11_1010
59	PC6	0b11_1011
60	PC5	0b11_1100
61	PC4	0b11_1101
62	PC1	0b11_1110
63	PC0	0b11_1111

Note that the interrupt vector table differs from the interrupt priority table in only two ways:

- FCC and SCC vectors are fixed; they are not affected by the SCC group mode, spread mode, or the relative priority order of the FCCs and SCCs.
- An error vector exists as the last entry in Table 22-2. The error vector is issued when no interrupt is requesting service.

22.4.1 Port C External Interrupts

There are 16 external interrupts, coming from the parallel I/O port C pins, PC[0:1, 4:15, 23, 29]. When one of these pins is configured as an input, a change according to the CPM external interrupt control register (SIEXR) causes an interrupt request signal to be sent to the interrupt controller. PC[0:1, 4:15, 23, 29] lines can be programmed to assert an interrupt request upon any change. Each port C line asserts a unique interrupt request to the interrupt pending register and has a different internal interrupt priority level within the interrupt controller.

Requests can be masked independently in the interrupt mask register (SIMR). Notice that the global SIMR is cleared on system reset so pins left floating do not cause false interrupts.

22.5 CPM Interrupt Programming Model

The interrupt controller registers control configuration, prioritization, and masking of interrupts. They also include registers for determining the interrupt sources. These registers are described in Section 22.5.1, “Interrupt Controller Registers.”

22.5.1 Interrupt Controller Registers

There are seven interrupt controller registers, described in the following sections:

- Section 22.5.1.1, “CPM Interrupt Configuration Register (SICR)”
- Section 22.5.1.2, “CPM Interrupt Priority Registers (SCPRR_H, SCPRR_L)”
- Section 22.5.1.3, “CPM Interrupt Pending Registers (SIPNR_H, SIPNR_L)”
- Section 22.5.1.4, “CPM Interrupt Mask Registers (SIMR_H, SIMR_L)”
- Section 22.5.1.5, “CPM Interrupt Vector Register (SIVEC)”
- Section 22.5.1.6, “CPM External Interrupt Control Register (SIEXR)”

22.5.1.1 CPM Interrupt Configuration Register (SICR)

The CPM interrupt configuration register (SICR), shown in Figure 22-3, defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table, Table 22-1.

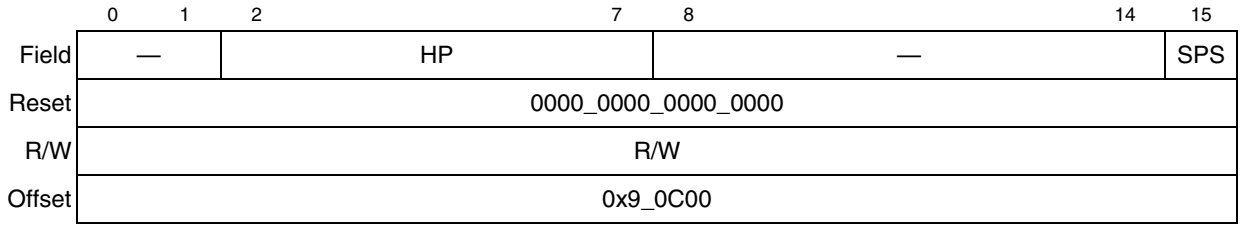


Figure 22-3. CPM Interrupt Configuration Register (SICR)

The SICR register bits are described in Table 22-3.

Table 22-3. SICR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–7	HP	Highest priority. Specifies the 6-bit interrupt number of the single interrupt controller interrupt source that is advanced to the highest priority in the table. HP can be modified dynamically. To retain the original priority, program HP to the interrupt number assigned to the interrupt request with the highest priority.
8–14	—	Reserved, should be cleared.
15	SPS	Spread priority scheme. Selects the relative YCC priority scheme. It cannot be changed dynamically. 0 Grouped. The YCCs are grouped by priority at the top of the table. 1 Spread. The YCCs are spread by priority in the table.

22.5.1.2 CPM Interrupt Priority Registers (SCPRR_H, SCPRR_L)

The CPM high interrupt priority register (SCPRR_H), shown in Figure 22-4, defines priorities between the FCCs.

	0–2	3–5	6–8	9–11	12–15
Field	XC1P	XC2P	XC3P	XC4P	—
Reset	000	001	010	011	000
R/W	R/W				
Offset	0x9_0C14				

	16–18	19–21	22–24	25–27	28–31
Field	XC5P	XC6P	XC7P	XC8P	—
Reset	100	101	110	111	000
R/W	R/W				
Offset	0x9_0C16				

Figure 22-4. CPM High Interrupt Priority Register (SCPRR_H)

Table 22-4 describes SCPRR_H fields.

Table 22-4. SCPRR_H Field Descriptions

Bits	Name	Description
0–2	XC1P	Priority order. Defines which FCC asserts its request in the XCC1 priority position. The user should not program the same FCC to more than one priority position (1–8). These bits can be changed dynamically. 000 FCC1 asserts its request in the XCC1 position. 001 FCC2 asserts its request in the XCC1 position. 010 Reserved 011 XCC1 position not active 100 Reserved 101 Reserved 110 XCC1 position not active 111 XCC1 position not active
3–11, 16–27	XC2P–XC8P	Same as XC1P, but for XCC2–XCC8
13–15, 28–31	—	Reserved, should be cleared.

The CPM low interrupt priority register (SCPRR_L), shown in Figure 22-5, defines the prioritization of the SCCs.

	0–2	3–5	6–8	9–11	12–15
Field	YC1P	YC2P	YC3P	YC4P	—
Reset	000	001	010	011	0000
R/W	R/W				
Offset	0x9_0C18				

	16–18	19–21	22–24	25–27	28–31
Field	YC5P	YC6P	YC7P	YC8P	—
Reset	100	101	110	111	0000
R/W	R/W				
Offset	0x9_0C20				

Figure 22-5. CPM Low Interrupt Priority Register (SCPRR_L)

Table 22-5 describes SCPRR_L fields.

Table 22-5. SCPRR_L Field Descriptions

Bits	Name	Description
0–2	YC1P	Priority order. Defines which SCC asserts its request in the YCC1 priority position. Do not program the same SCC to multiple priority positions. This field can be changed dynamically. 000 SCC1 asserts its request in the YCC1 position. 001 Reserved 010 SCC3 asserts its request in the YCC1 position. 011 SCC4 asserts its request in the YCC1 position. 100 Reserved 101 Core interface asserts its request in the YCC1 position 110 System SDMA asserts its request in the YCC1 position 111 Local SDMA asserts its request in the YCC1 position
3–11, 16–27	YC2P–YC8P	Same as YC1P, but for YCC2–YCC8
12–15, 28–31	—	Reserved, should be cleared.

22.5.1.3 CPM Interrupt Pending Registers (SIPNR_H, SIPNR_L)

Each bit in the CPM interrupt pending registers (SIPNR_H and SIPNR_L), shown in Figure 22-6 and Figure 22-7, corresponds to an interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PC0	PC1	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC23	PC29
Reset	Undefined (The user should write 1s to clear these bits before using.)															
R/W	R/W															
Offset	0x9_0C08															

	16	31
Field	—	
Reset	Undefined (The user should write 1s to clear these bits before using.)	
R/W	R/W	
Offset	0x9_0C10	

Figure 22-6. SIPNR_H Fields

Figure 22-7 shows SIPNR_L fields.

	0	1	2–7	8	9	10	11	12	13	14	15
Field	FCC1	FCC2	—	SCC1	—	SCC3	SCC4	—	Core I/F	SDMA SYS	SDMA LCL
Reset	0000_0000_0000_0000[1]										
R/W	R/W										
Offset	0x9_0C0C										

	16	17	18	19	20	21–25	26	27	28	29	30	31
Field	I^2C	SPI	RTT	SMC1	SMC2	—	USB	TIMER1	TIMER2	TIMER3	TIMER4	—
Reset	0000_0000_0000_0000[1]											
R/W	R/W											
Offset	0x9_0C0E											

[1] These fields are zero after reset because their corresponding mask register bits are cleared (disabled).

Figure 22-7. SIPNR_L Fields

When a pending interrupt is handled, the user clears the corresponding SIPNR bit. However, if an event register exists, the unmasked event register bits should be cleared instead, causing the SIPNR bit to be cleared.

SIPNR bits are cleared by writing ones to them. Because the user can only clear bits in this register, writing zeros to this register has no effect.

Note that the SCC/FCC SIPNR bit positions are not changed according to their relative priority.

22.5.1.4 CPM Interrupt Mask Registers (SIMR_H, SIMR_L)

Each bit in the CPM interrupt mask register (SIMR) corresponds to a interrupt source. The user masks an interrupt by clearing and enables an interrupt by setting the corresponding SIMR bit. When a masked

interrupt occurs, the corresponding SIPNR bit is set, regardless of the SIMR bit although no interrupt request is passed to the core.

If an interrupt source requests interrupt service when the user clears its SIMR bit, the request stops. If the user sets the SIMR bit later, a previously pending interrupt request is processed by the core, according to its assigned priority. The SIMR can be read by the user at any time.

Figure 22-8 shows the SIMR_H register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PC0	PC1	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC23	PC29
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0C1C															

	16	31
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Offset	0x9_0C1E	

Figure 22-8. SIMR_H Register

Figure 22-9 shows SIMR_L.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FCC1	FCC2	—						SCC1	—	SCC3	SCC4	—	Core I/F	SDMA SYS	SDMA LCL
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0C20															

	16	17	18	19	20	21	25	26	27	28	29	30	31
Field	I^2C	SPI	RTT	SMC1	SMC2	—		USB	TIMER1	TIMER2	TIMER3	TIMER4	—
Reset	0000_0000_0000_0000												
R/W	R/W												
Offset	0x9_0C22												

Figure 22-9. SIMR_L Register

Note the following:

- SCC/FCC SIMR 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 SIMR 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 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.

22.5.1.5 CPM Interrupt Vector Register (SIVEC)

The CPM interrupt vector register (SIVEC), shown in Figure 22-10, contains an 8-bit code representing the unmasked interrupt source of the highest priority level.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Interrupt Code						0	0	0	0	0	0	0	0	0	0
Reset	0000_0000_0000_0000															
R/W	R															
Offset	0x9_0C04															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0000_0000_0000_0000															
R/W	R															
Offset	0x9_0C06															

Figure 22-10. CPM Interrupt Vector Register (SIVEC)

The SIVEC can be read as either a byte, half word, or word. When read as a byte, a branch table can be used in which each entry contains one instruction (branch). When read as a half word, each entry can contain a full routine of up to 256 instructions. The interrupt code is defined such that its two lsbs are zeros, allowing indexing into the table, as shown in Figure 22-11.

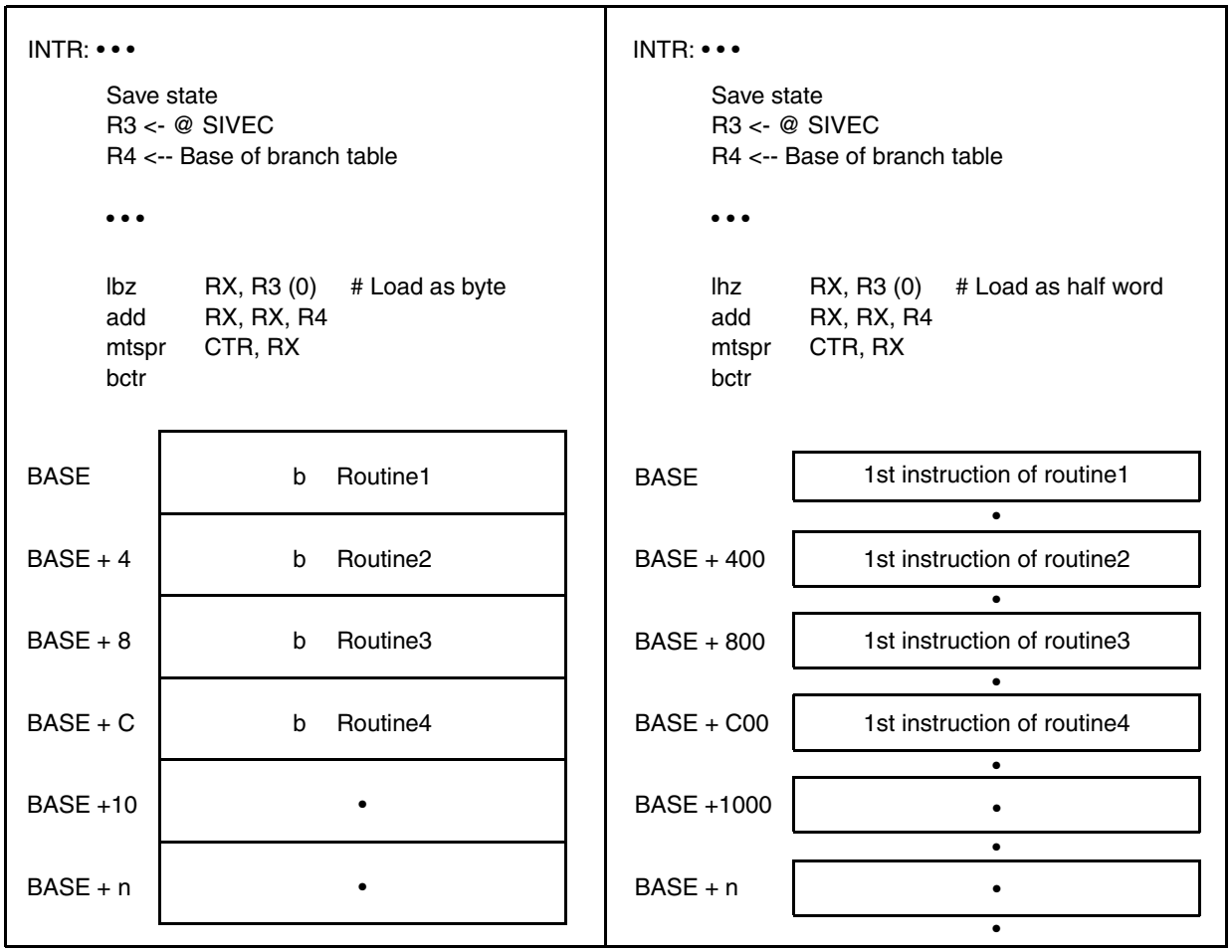


Figure 22-11. Interrupt Table Handling Example

22.5.1.6 CPM External Interrupt Control Register (SIEXR)

Each defined bit in the CPM external interrupt control register (SIEXR), shown in Figure 22-12, determines whether the corresponding port C line asserts an interrupt request upon either a high-to-low change or any change on the pin. External interrupts can come from port C (PC[0:15]).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	EDPC 0	EDPC 1	EDPC 4	EDPC 5	EDPC 6	EDPC 7	EDPC 8	EDPC 9	EDPC 10	EDPC 11	EDPC 12	EDPC 13	EDPC 14	EDPC 15	EDPC 23	EDPC 29
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0C24															

	16	31
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	R
Offset	0x9_0C26	

Figure 22-12. CPM External Interrupt Control Register (SIEXR)

Table 22-6 describes SIEXR fields.

Table 22-6. SIEXR Field Descriptions

Bits	Name	Description
0–15	EDPCx	Edge detect mode for port Cx. The corresponding port C line (PCx) asserts an interrupt request according to the following: 0 Any change on PCx generates an interrupt request. 1 High-to-low change on PCx generates an interrupt request
16–31	—	Reserved

Chapter 23
Serial Interface with Time-Slot Assigner

23.1 Overview

Figure 23-1 shows a block diagram of the time-slot assigner (TSA). One serial interface (SI) block in the MPC8555E (SI2) can be programmed to handle three TDM lines concurrently. The TDM channels on SI2 are referred to as TDMa2, TDMb2, and TDMc2.

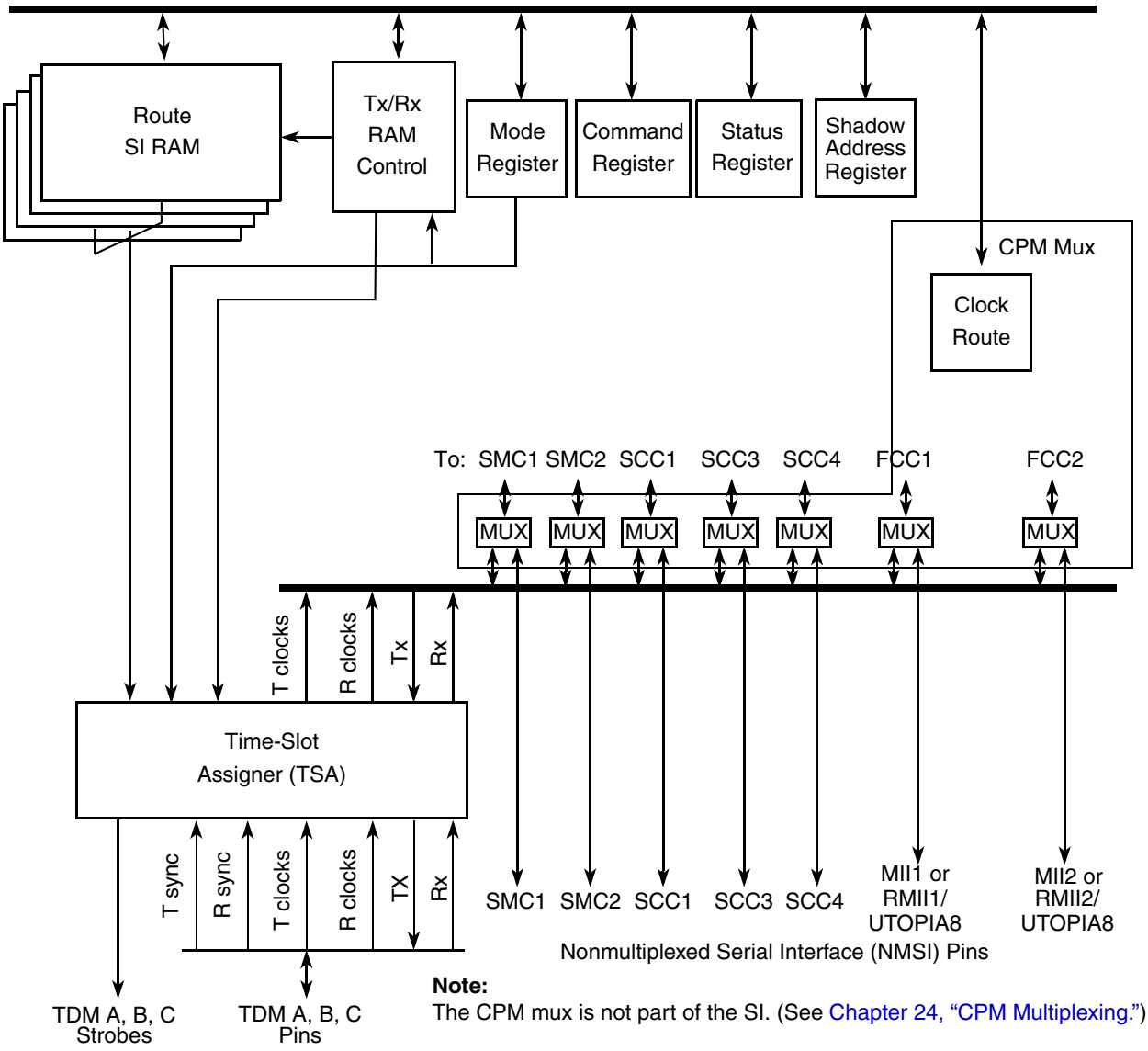


Figure 23-1. SI Block Diagram

If the TSA is not used as intended, it can be used to generate complex wave forms on dedicated output pins. For instance, it can program these pins to implement stepper motor control or variable-duty cycle and period control on-the-fly.

23.2 Features

SI has the following features:

- Can connect to three independent TDM channels. Each TDM can be one of the following:
  - T1 or E1 line
  - Integrated services digital network primary rate (PRI)
  - An ISDN basic rate–interchip digital link (IDL) channel in up to three TDM channels—each IDL channel requires support from a separate SCC
  - ISDN basic rate–general circuit interface (GCI) in up to two TDM channels—each GCI channel requires support from a separate SMC
  - E3 or DS3 clear channel
  - User-defined interfaces
- Independent, programmable transmit and receive routing paths
- Independent transmit and receive frame syncs allowed
- Independent transmit and receive clocks allowed
- Selection of rising/falling clock edges for the frame sync and data bits
- Supports 1× and 2× input clocks (1 or 2 clocks per data bit)
- Selectable delay (0–3 bits) between frame sync and frame start
- Four programmable strobe outputs and two clock output pins
- 1- or 8-bit resolution in routing, masking, and strobe selection
- Supports frames up to 16,384 bits long
- Internal routing and strobe selection can be dynamically programmed
- Supports automatic echo and loopback mode for each TDM
- Maximum TDM frequency is serial-dependent:
  - For FCCs transparent $\frac{\text{CPM Clock}}{4}$
  - For FCCs HDLC
  - For all other serials $\frac{\text{CPM Clock}}{3}$

NOTE

This clock ratio is based on the hardware architecture and does not ensure that an application will run at that speed. It is the responsibility of the system designer to check AC specifications of the I/O pins and determine the maximum frequency.

23.3 Overview

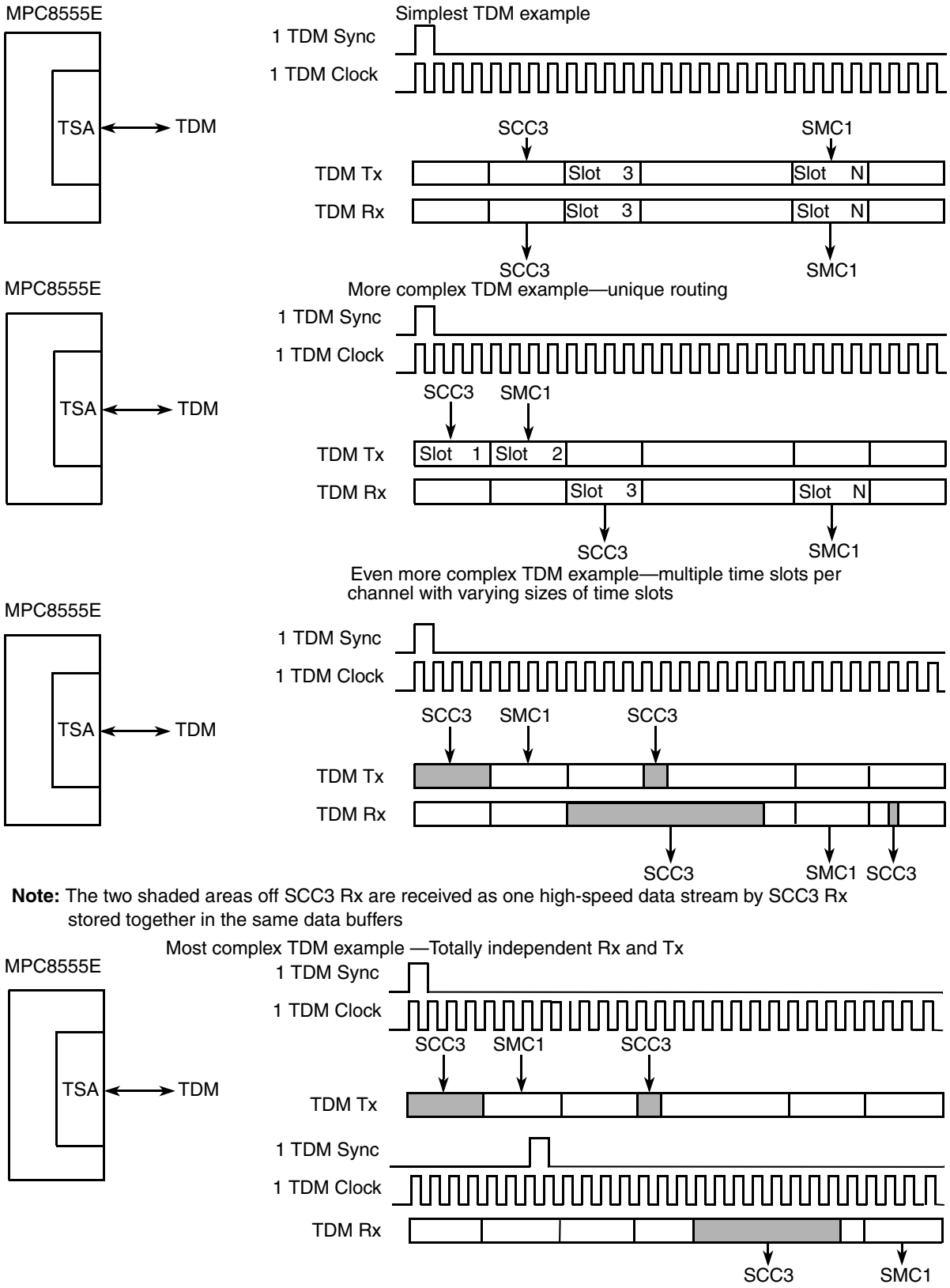
The TSA implements both internal route selection and time-division multiplexing (TDM) for multiplexed serial channels. The TSA supports the serial bus rate and format for most standard TDM buses, including T1 and E1 highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates. The two popular ISDN basic rate buses (interchip digital link (IDL) and general-circuit interface (GCI), also known as IOM-2) are supported.

Because the SI supports three TDMs, it is possible to simultaneously support a combination of three T1 or E1 lines, and basic rate or primary rate ISDN channels.

The TDM channel can support E3 or DS-3 rates as a clear channel in a serial interface (clock ratio 1/4).

TSA programming is independent of the protocol used. The serial controllers can be programmed for any synchronous protocol without affecting TSA programming. The TSA simply routes programmed portions of the received data frame from the TDM pins to the target controller, while the target controller handles the received data in the actual protocol.

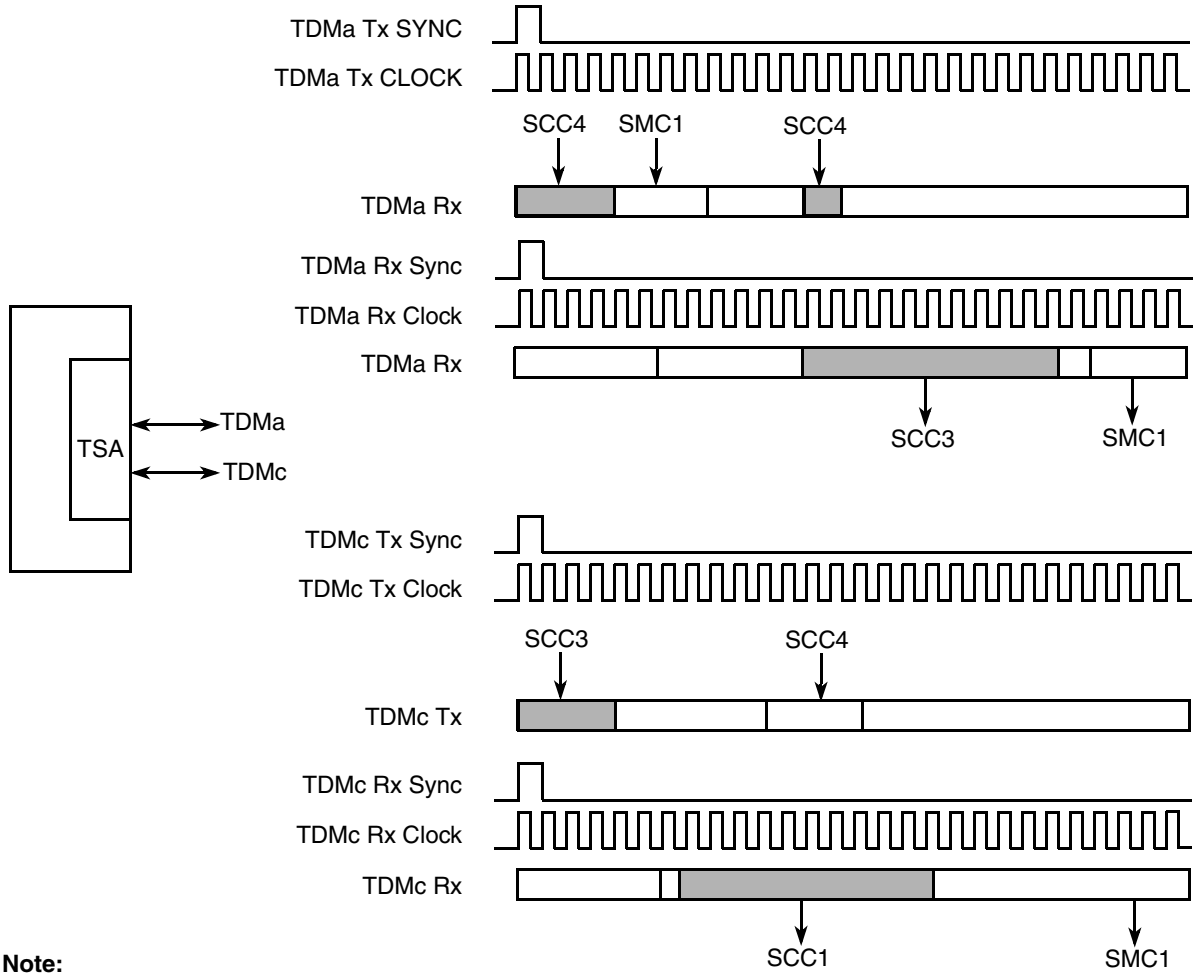
In its simplest mode, the TSA identifies the frame using one sync pulse and one clock signal provided externally by the user. This can be enhanced to allow independent routing of the receive and transmit data on the TDM. Additionally, the definition of a time-slot need not be limited to 8 bits or even to a single contiguous position within the frame. Finally, the user can provide separate receive and transmit syncs as well as clocks. Figure 23-2 shows example TSA configurations ranging from the simplest to the most complex.



Note: The two shaded areas off SCC3 Rx are received as one high-speed data stream by SCC3 Rx stored together in the same data buffers

Figure 23-2. Various Configurations of a Single TDM Channel

At its most flexible, the TSA can provide three separate TDM channels, each with independent receive and transmit routing assignments and independent sync pulse and clock inputs. Thus, the TSA can support six, independent, half-duplex TDM sources, three in reception and three in transmission, using six sync inputs and six clock inputs. Figure 23-3 shows a dual-channel example.



Note:
SCCs can receive on one TDM and transmit on another (SCC1 and SCC3).

Figure 23-3. Dual TDM Channel Example

In addition to channel programming, the TSA supports up to four strobe outputs that may be asserted on a bit or byte basis. These strobes are completely independent from the channel routing used by the SCCs and SMCs. The strobe outputs are useful for interfacing to other devices that do not support the multiplexed interface or for enabling/disabling three-state I/O buffers in a multiple-transmitter architecture. Notice that open-drain programming on the TXDx pins that supports a multiple-transmitter architecture occurs in the parallel I/O block. These strobes can also be used for generating output wave forms to support such applications as stepper-motor control.

Most TSA programming is done in the two 256 × 16-bit SI2 RAMs. These SI2 RAMs are directly accessible by the core in the internal register section of the MPC8555E and are not associated with the dual-port RAM. One SI2 RAM is always used to program the transmit routing; the other is always used to

program the receive routing. SI2 RAMs can be used to define the number of bits/bytes to be routed to the FCC, SCC, or SMC and determine when external strobes are to be asserted and negated.

The size of the SI2 RAM available for time-slot programming depends on the user's configuration. The user defines how many of the 256 entries are related to each TDM. The resolution of the division is by fractions of 32. If on-the-fly changes are allowed, the SI2 RAM entries are reduced according to the user's programming. The maximum frame length that can be supported in any configuration is 16,384 bits.

The maximum external serial clock that may be an input to the TSA is CPM CLK/3.

The SI supports two testing modes—echo and loopback.

- The echo mode provides a return signal from the physical interface by retransmitting the signal it has received. The physical interface echo mode differs from the individual FCC or SCC echo mode in that it can operate on the entire TDM signal rather than just on a particular serial channel.
- Loopback mode causes the physical interface to receive the same signal it is sending. The SI loopback mode checks more than the individual serial loopback; it checks both the SI and the internal channel routes.

NOTE

The flexibility described in the preceding section can be applied to each of the three TDM channels and to all serial interfaces independently.

23.4 Enabling Connections to TSA

Each serial interface can be independently enabled to connect to one of the following: TSA, UTOPIA, MII, or dedicated external pins. Note the following:

- Each FCC can be connected to a dedicated MII to or one of the three TDMs, or to an 8-bit UTOPIA level II interface.
- Each SCC or SMC can be connected to one of the three TDMs or to its own set of pins.

The three TDMs are connected to three independent TDM interfaces. Figure 23-4 illustrates the connection between the TSA and the serial interfaces. The connection is made by programming the CPM mux. See Chapter 24, "CPM Multiplexing." After the connections are made, the exact routing decisions are made in the SI2 RAM.

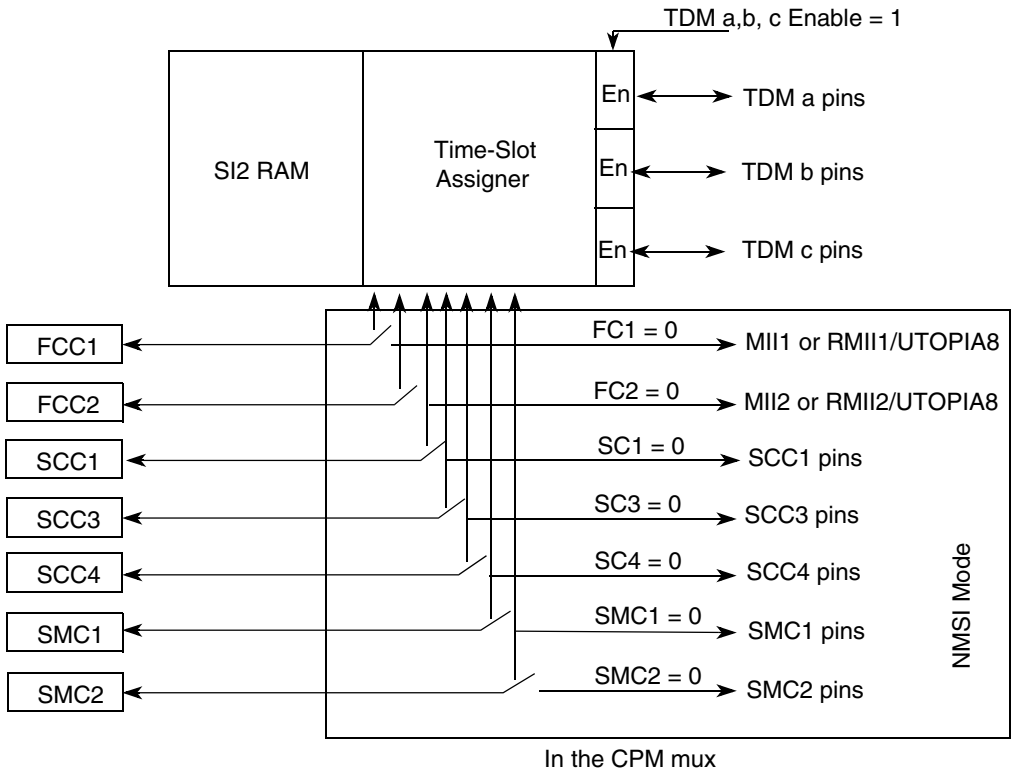


Figure 23-4. Enabling Connections to the TSA

23.5 Serial Interface RAM

The SI has a transmit RAM and a receive RAM, each with four banks of 64 half-word entries that enable it to control TDM channel routing to all serial devices. The SI2 RAMs are uninitialized after power-on reset; unwanted results can occur if the user does not program them before enabling the multiplexed channels.

Each 16-bit SI RAM entry defines the routing of 1–8 bits or bytes at a time. In addition to the routing, up to four strobe pins (logic OR of four strobes in the transmit RAM and four in receive RAM) can be asserted according to the programming of the RAMs. The four SI2 RAM banks can be configured in many different ways to support various TDM channels. The user can define the size of each SI2 RAM that is related to a certain TDM channel by programming the starting bank of that TDM. Programming the starting shadow bank address, described in Section 23.6.3, “SI2 RAM Shadow Address Registers (SI2RSR),” determines whether this RAM has a shadow for changing SI2 RAM entries while the TDM channel is active. This reduces the number of available SI2 RAM entries for that TDM.

23.5.1 One Multiplexed Channel with Static Frames

The example in Figure 23-5 shows one of many possible settings. With this configuration, the SI2 RAM has 256 entries for transmit data and strobe routing and 256 entries for receive data and strobe routing. This configuration should be chosen only when one TDM is required and the routing on that TDM does not

need to be dynamically changed. The number of entries available in the SI2 RAM is determined by the user.

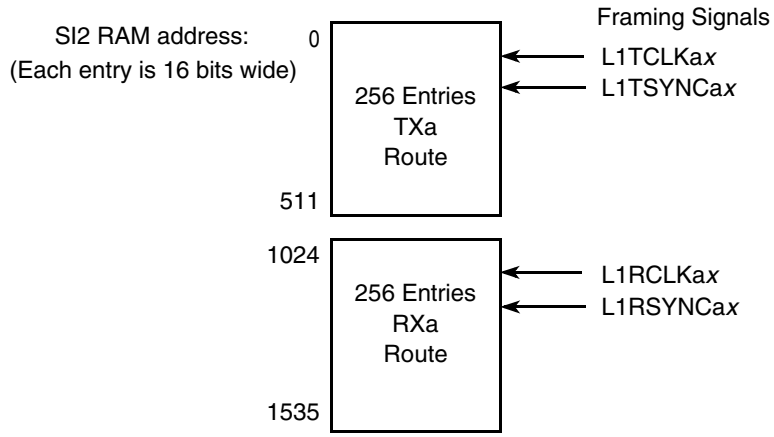


Figure 23-5. One TDM Channel with Static Frames and Independent Rx and Tx Routes

23.5.2 One Multiplexed Channel with Dynamic Frames

In the configuration shown in Figure 23-5, one multiplexed channel has 256 entries for transmit data and strobe routing and 256 entries for receive data and strobe routing. Each RAM has two sections, the current-route RAM and a shadow RAM for changing serial routing dynamically.

After programming the shadow RAM, the user sets SI2CMDR[CSR*xn*] for the associated channel. When the next frame sync arrives, the SI automatically exchanges the current-route RAM for the shadow RAM. See Section 23.5.5, “Static and Dynamic Routing.”

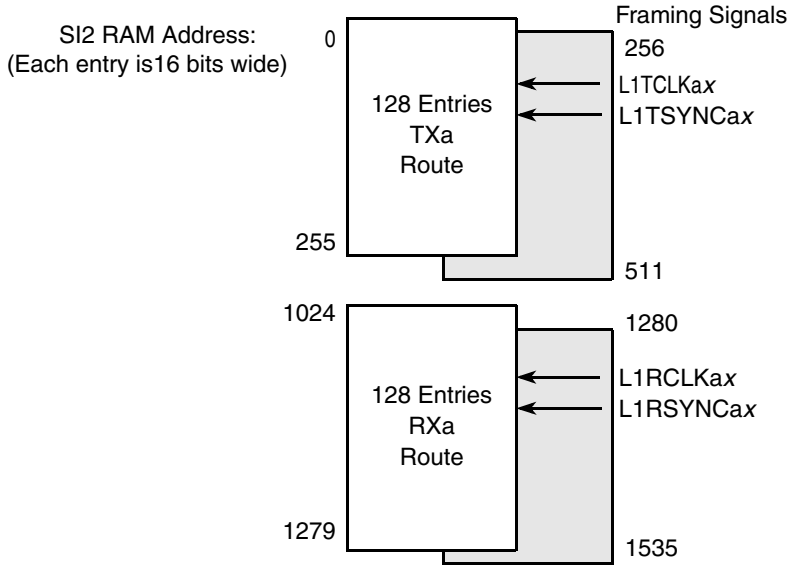


Figure 23-6. One TDM Channel with Shadow RAM for Dynamic Route Change

This configuration should be chosen when only one TDM is needed, but dynamic rerouting may be needed on that TDM. Similarly, for three TDM channels, the number of SI2 RAM entries are reduced for every TDM channel programmed for shadow mode.

23.5.3 Programming SI2 RAM Entries

The programming of each entry in the SI2 RAM determines the routing of the serial bits (or bit groups) and the assertion of strobe outputs. Figure 23-7 shows the entry fields.

	0	1	2	3	4	5	6	7 10	11 13	14	15
Field	—	SWTR	SSEL1	SSEL2	SSEL3	SSEL4	—	CSEL	CNT	BYT	LST
R/W	R/W										
Offset	See Chapter 21, “Communications Processor Module Overview.”										

Figure 23-7. SI2 RAM Entry Fields

The SI2 RAM entry fields function as described in Table 23-1.

Table 23-1. SI2 RAM Entry

Bits	Name	Description
0	—	Reserved, should be cleared.
1	SWTR	Switch Tx and Rx. Valid only in the receive route RAM and ignored in the transmit route RAM. SWTR affects the operation of both L1RXD and L1TXD. SWTR is set only in special situations where the user prefers to receive data from a transmit pin and transmit data on a receive pin. For instance, where devices A and B are connected to the same TDM, each with different time-slots. Normally, there is no opportunity for stations A and B to communicate with each other directly over the TDM, because they both receive the same TDM receive data and transmit on the same TDM transmit signal. 0 Normal operation of L1TXD and L1RXD 1 Data for this entry is sent on L1RXD and received from L1TXD See Figure 23-8 for details.
2–5	SSELx	Strobe select. There are four strobes available that can be assigned to the receive RAM and asserted/negated with the received clock of this TDM channel (L1RCLKx). They can also be assigned to the transmit RAM and asserted/negated with the transmit clock of this TDM channel (L1TCLKx). Each bit corresponds to the value the strobe should have during this bit/byte group. There are four strobe pins for all eight strobe bits in the SI2 RAM entries, so the value on a strobe pin is the logical OR of the Rx and Tx RAM entry strobe bits. Multiple strobes can be asserted simultaneously. A strobe configured to be asserted in consecutive SI2 RAM entries remains continuously asserted for both entries. A strobe asserted on the last entry in a table is negated after the last entry is processed. **Note:** Each strobe is changed with the corresponding RAM clock and is output only if the corresponding parallel I/O is configured as a dedicated pin. If a strobe is programmed to be asserted in more than one set of entries (the SI route entries for more then one TDM channel select the same strobe), the assertion of the strobe corresponds to the logical OR of all possible sources. This use of strobes is not useful for most applications. A given strobe should be selected in only one set of SI2 RAM entries.
6	—	Reserved, should be cleared.

Table 23-1. SI2 RAM Entry (continued)

Bits	Name	Description
7–10	CSEL	Channel select. Determines the routing of the bit/byte group. 0000 The bit/byte group is not supported by the MPC8555E. The transmit data pin is three-stated and the receive data pin is ignored. 0001 Routed to SCC1 0010 Reserved 0011 Routed to SCC3 0100 Routed to SCC4 0101 Routed to SMC1 0110 Routed to SMC2 0111 The bit/byte group is not supported by the MPC8555E. This code is also used in SCIT mode as the D channel grant. See Section 23.8.2.2, “SCIT Programming.” 1000 Reserved 1001 Routed to FCC1 1010 Routed to FCC2 1011 Reserved 11xx Reserved
11–13	CNT	Count. Indicates the number of bits/bytes (according to the BYT bit) that the routing and strobe select of this entry controls. 000 = 1 bit/byte; 111= 8 bits/bytes.
14	BYT	Byte resolution 0 Bit resolution. The CNT value indicates the number of bits in this group. 1 Byte resolution. The CNT value indicates the number of bytes in this group.
15	LST	Last entry in the RAM. Whenever SI2 RAM is used, LST must be set in one of the Tx or Rx entries of each group. Even if all entries of a group are used, this bit must still be set in the last entry. 0 Not the last entry in this section of the route RAM. 1 Last entry in this RAM. After this entry, the SI waits for the sync signal to start the next frame. **Note:** There must be only an even number of entries in an SI2 RAM frame, because LST is active only in odd-numbered entries (assuming the entry count starts with 0). Therefore, to obtain an even number of entries, an entry may need to be split into two entries. Also note that, to avoid errors in switching to and from shadow SI RAM, the last entry in SI RAM should not be programmed to 1-bit resolution (that is, CNT = 000 and BYT = 0).

Figure 23-8 shows how SWTR can be used.

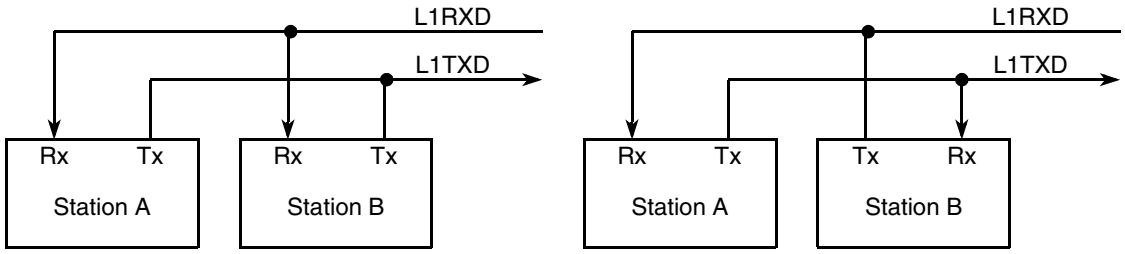


Figure 23-8. Using the SWTR Feature

The SWTR option lets station B listen to transmissions from station A and send data to station A. To do this, station B would set SWTR in its receive route RAM. For this entry, receive data is taken from the L1TXD pin and data is sent on the L1RXD pin. If the user wants to listen only to station A transmissions

and not send data on L1RXD, the CSEL bits in the corresponding transmit route RAM entry should be cleared to prevent transmission on the L1RXD pin.

Station B can transmit data to station A by setting the SWTR bit of the entry in its receive route RAM. Data is sent on L1RXD rather than L1TXD, according to the transmit route RAM. Note that this configuration could cause collisions with other data on L1RXD unless an available (quiet) time slot is used. To transmit on L1RXD and not receive data on L1TXD, clear the CSEL bits in the receive route RAM.

NOTE

If the transmit and receive sections of the TDM do not use a common clock source, the SWTR feature can cause erratic behavior.

23.5.4 SI2 RAM Programming Example

This example shows how to program the RAM to support the 10-bit IDL bus. Figure 23-23 shows the 10-bit IDL bus format. In this example, the TSA supports the B1 channel with SCC3, the D channel with SCC1, the first 4 bits of the B2 channel with an external device (using a strobe to enable the external device), and the last 4 bits of B2 with SMC1. Additionally, the TSA marks the D channel with another strobe signal.

First, divide the frame from the start (the sync) to the end of the frame according to the support that is required:

- 8 bits (B1)—SCC3
- 1 bit (D)—SCC1 + strobe 1
- 1 bit—no support
- 4 bits (B2)—strobe 2
- 4 bits (B2)—SMC1
- 1 bit (D)—SCC1 + strobe 1

Each of these six divisions can be supported by a single SI2 RAM entry. Thus, six SI2 RAM entries are needed. See Table 23-2.

Table 23-2. SI2 RAM Entry Descriptions

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0000	0011	000	1	0	8-bit SCC3
1	0	1000	0001	000	0	0	1-bit SCC1 strobe1
2	0	0000	0000	000	0	0	1-bit no support
3	0	0100	0000	011	0	0	4-bit strobe2
4	0	0000	0101	011	0	0	4-bit SMC1
5	0	1000	0001	000	0	1	1-bit SCC1 strobe1

NOTE

IDL requires the same routing for both receive and transmit., Therefore, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI2 RAM. Then SI2MR[CRTx] can be used to instruct the SI2 RAM to use the same clock and sync to simultaneously control both sets of SI2 RAM entries.

23.5.5 Static and Dynamic Routing

The SI2 RAM has two operating modes for the TDMs:

- Static routing. The number of SI2 RAM entries is determined by the banks the user relates to the corresponding TDM and is divided into two parts (Rx and Tx). The following sequence must be followed to program the routing entries.
  - All serial devices connected to the TSA must be disabled.
  - SI routing can be modified.
  - All appropriate serial devices connected to the TSA must be re-enabled.
- Dynamic routing. A TDM's routing definition can be modified while FCCs, SCCs, or SMCs are connected to the TDM. The number of SI2 RAM entries is determined by the banks the user relates to the corresponding TDM channel and is divided into four parts (Rx, Rx shadow, Tx, and Tx shadow).

Dynamic changes divide portions of the SI2 RAM into current-route and shadow RAM. Once the current-route RAM is programmed, the TSA and SI channels are enabled, and TSA operation begins. When a change in routing is required, the shadow RAM must be programmed with the new route and SI2CMDR[CSR*xn*] must be set. As a result, as soon as the corresponding sync arrives the SI exchanges the shadow RAM with the current-route RAM and resets CSR*xn* to indicate that the operation is complete. At this time, the user may change the routing again. Notice that the original current-route RAM is now the shadow RAM and vice versa. Figure 23-9 shows an example of the shadow RAM exchange process for two TDM channels both with half of the RAM as a shadow.

If for instance one TDM with dynamic changes is programmed to own all four banks, and the shadow is programmed to the last two banks, the initial current-route RAM addresses in the SI2 RAM are as follows.

- 0–255: TXa route
- 1024–1279: RXa route

The initial shadow RAMs are at addresses:

- 256–511: TXa route
- 1280–1535: RXa route

The user can read any RAM at any time, but for proper SI operation the user must not attempt to write the current-route RAM. The SI2 status register (SI2STR) can be read to find out which part of the RAM is the current-route RAM. The user can also externally connect one of the strobes to an interrupt pin to generate an interrupt on a particular SI2 RAM entry starting or ending execution by the TSA.

NOTE

The current-route and shadow SI RAMs of a given TDM*x* should be contiguous; that is, the current-route and shadow SI RAMs of differing TDM*x* should not be interleaved. An example is shown in Figure 23-9.

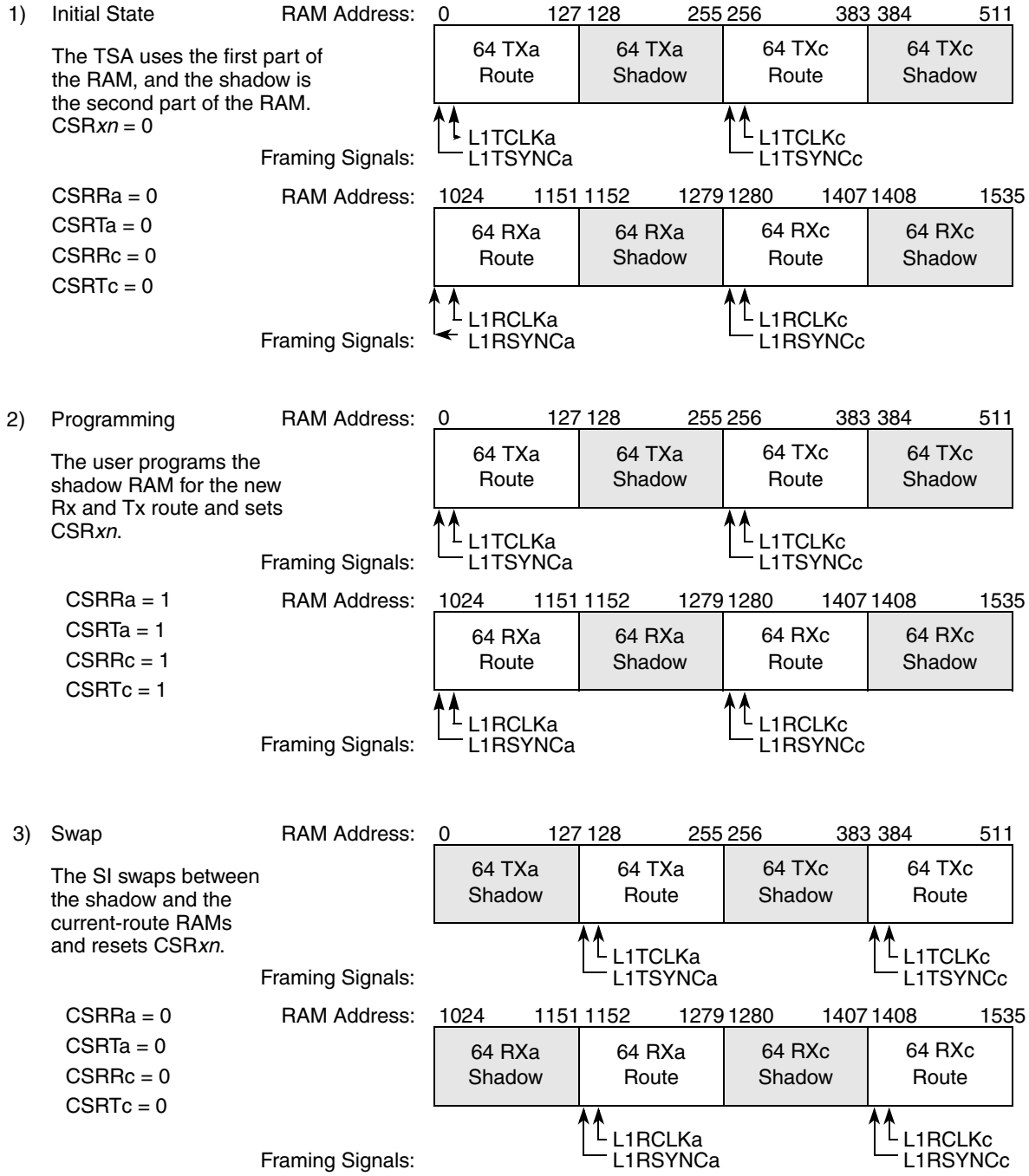


Figure 23-9. Example: SI2 RAM Dynamic Changes, TDMa and TDMc, Same SI2 RAM Size

23.6 Serial Interface Registers

The serial interface registers are described in the following sections.

23.6.1 SI Global Mode Registers (SI2GMR)

The SI global mode registers (SI2GMR), shown in Figure 23-10, defines the activation state of the TDM channels for SI2.

	0	1	2	3	4	5	6	7
Field	—	STZC	STZB	STZA	—	ENC	ENB	ENA
Reset	0000_0000							
R/W	R/W							
Offset	0x0x9_1B48 (SI2GMR)							

Figure 23-10. SI Global Mode Registers (SI2GMR)

Table 23-3 describes SI2GMR.

Table 23-3. SI2GMR Field Descriptions

Bit	Name	Description
0,4,	—	Reserved, should be cleared.
1, 2, 3	STZx	Program L1TXDx to zero for TDM a, b, or c 0 Normal operation 1 L1TXDx = 0 until serial clocks are available, which is useful for GCI activation. See Section 23.8.1, “SI GCI Activation/Deactivation Procedure.”
5, 6, 7	ENx	Enable TDMx. Note that enabling a TDM is the last step in initialization. 0 TDM channel x is disabled. The SI2 RAMs and routing for TDMx are in a state of reset, but all other SI functions still operate. 1 All TDMx functions are enabled.

23.6.2 SI Mode Registers (SI2MR)

There are three SI mode registers (SI2MR), shown in Figure 23-11, one for each TDM channel (SI2AMR, SI2BMR, and SI2CMR). They are used to define SI operation modes and allow the user (with SI2 RAM) to support any or all of the ISDN channels independently when in IDL or GCI mode. Any extra serial channel can then be used for other purposes.

	0	1 3	4 5	6 7	8	9	10	11	12	13	14 15
Field	—	SADx	SDMx	RFSDx	DSCx	CRTx	SLx	CEx	FEx	GMx	TFSDx
Reset	0000_0000_0000_0000										
R/W	R/W										
Offset	0x9_1B40 (SI2AMR), 0x9_1B42 (SI2BMR), 0x9_1B44 (SI2CMR)										

Figure 23-11. SI Mode Registers (SI2MR)

Table 23-4 describes SI2MR fields.

Table 23-4. SI2MR Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1–3	SADx	Starting bank address for the RAM of TDM a, b, or c. These three bits define the starting bank address of the SI2 RAM section that belongs to TDMx channel. **Note:** As noted previously, the SI2 RAM contains four banks of 64 entries for receive and four banks of 64 entries for transmit. The starting bank address of each TDM can be programmed with a granularity of 32 entries. The user can put the shadow RAM section of the same TDM on the same bank, but the user cannot put two different TDMs on the same bank. The last entry of a certain TDM is determined by the LST bit in the SI2 RAM entry. The user must set LST within the entries of SI2 RAM blocks for every TDM used, that is, before the starting address of the next TDM. 000 First bank, first 32 entries 001 First bank, second 32 entries 010 Second bank, first 32 entries 011 Second bank, second 32 entries 100 Third bank, first 32 entries 101 Third bank, second 32 entries 110 Fourth bank, first 32 entries 111 Fourth bank, second 32 entries
4–5	SDMx	SI Diagnostic Mode for TDM a, b, or c 00 Normal operation. 01 Automatic echo. In this mode, the TDM 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, $\overline{\text{L1RQ}x}$ 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 $\overline{\text{L1RQ}x}$ are inactive. This mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines. **Note:** In modes 01, 10, and 11, the receive and transmit clocks should be identical.
6–7	RFSDx	Receive frame sync delay for TDM a, b, or c. 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; use for GCI. 01 1-bit delay. Use for IDL 10 2-bit delay 11 3-bit delay Figure 23-12 and Figure 23-13 show how these bits are used.
8	DSCx	Double speed clock for TDM a, b, or c. Some TDMs, such as GCI, define the input clock to be twice as fast as the data rate and this bit controls this option. 0 The channel clock (L1RCLKx and/or L1TCLKx) is equal to the data clock. Use for IDL and most TDM formats. 1 The channel clock rate is twice the data rate. Use for GCI. **Note:** When an SI is in 2X mode (DSC = 1), the SI does not ignore sync signals asserted in the last phase of the last clock cycle of the frame.

Table 23-4. SI2MR Field Descriptions (continued)

Bits	Name	Description
9	CRTx	Common receive and transmit pins for TDM a, b, or c. Useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx can be used for their alternate functions. 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 and GCI. RFSD and TFSD are independent of one another in this mode.
10	SLx	Sync level for TDM a, b, or c. 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 TDM a, b, or c. The function depends on DSCx. When DSCx = 0: 0 The data is sent on the rising edge of the clock and received on the falling edge (use for IDL). 1 The data is sent on the falling edge of the clock and received on the rising edge. When DSCx = 1: 0 The data is sent on the rising edge of the clock and received on the rising edge. 1 The data is sent on the falling edge of the clock and received on the falling edge (use for GCI). See Figure 23-14 and Figure 23-15.
12	FEx	Frame sync edge for TDM a, b, or c. Determines whether L1RSYNCx and L1TSYNCx pulses are sampled with the falling/rising edge of the channel clock. See Figure 23-13, Figure 23-14, Figure 23-15, and Figure 23-16. 0 Falling edge. Use for IDL and GCI 1 Rising edge
13	GMx	Grant mode for TDM a, b, or c. 0 GCI/SCIT mode. The GCI/SCIT D channel grant mechanism for transmission is internally supported. The grant is one bit from the receive channel. This bit is marked by programming the channel select bits of the SI2 RAM with 0111 to assert an internal strobe on it. See Section 23.8.2.2, “SCIT Programming.” 1 IDL mode. A grant mechanism is supported if the corresponding CMXSCR[GR*x*] bit is set. The grant is a sample of L1GRx while L1RSYNCx is asserted. This grant mechanism implies the IDL access controls for transmission on the D channel. See Section 23.7.2, “IDL Interface Programming.”
14–15	TFSDx	Transmit frame sync delay for TDM a, b, or c. Determines the number of clock delays between the transmit sync and the first bit of the transmit frame. See Figure 23-16. 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

Figure 23-12 shows the one-clock delay from sync to data when *x*FSD = 01.

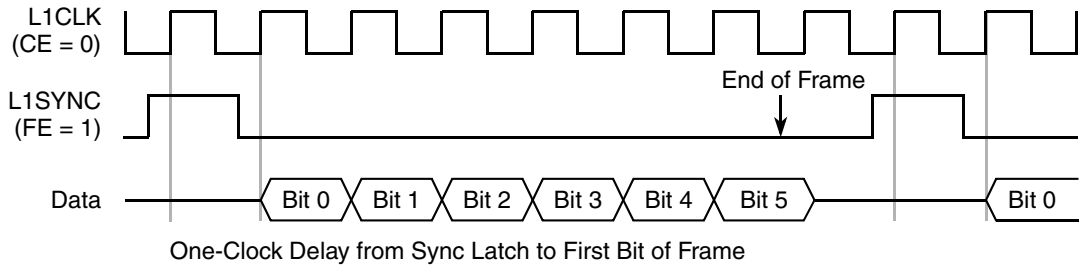


Figure 23-12. One-Clock Delay from Sync to Data (*x*FSD = 01)

Figure 23-13 shows the elimination of the single-clock delay shown in Figure 23-12 by clearing *x*FSD.

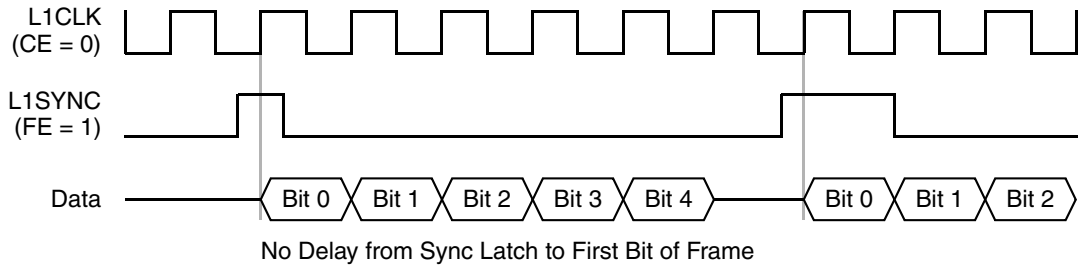


Figure 23-13. No Delay from Sync to Data (*x*FSD = 00)

Figure 23-14 shows the effects of changing FE when CE = 1 with a 1-bit frame sync delay.

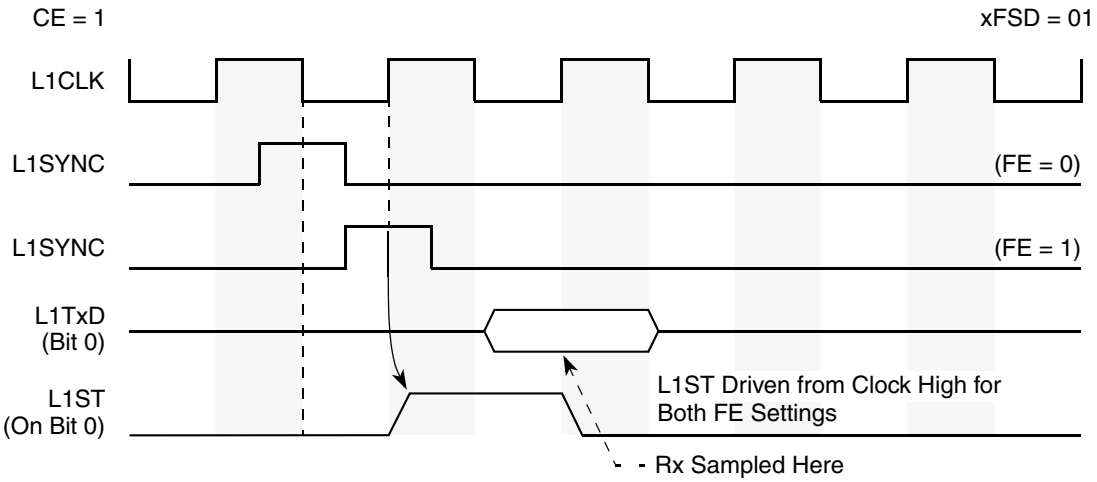


Figure 23-14. Falling Edge (FE) Effect when CE = 1 and *x*FSD = 01

Figure 23-15 shows the effects of changing FE when CE = 0 with a 1-bit frame sync delay.

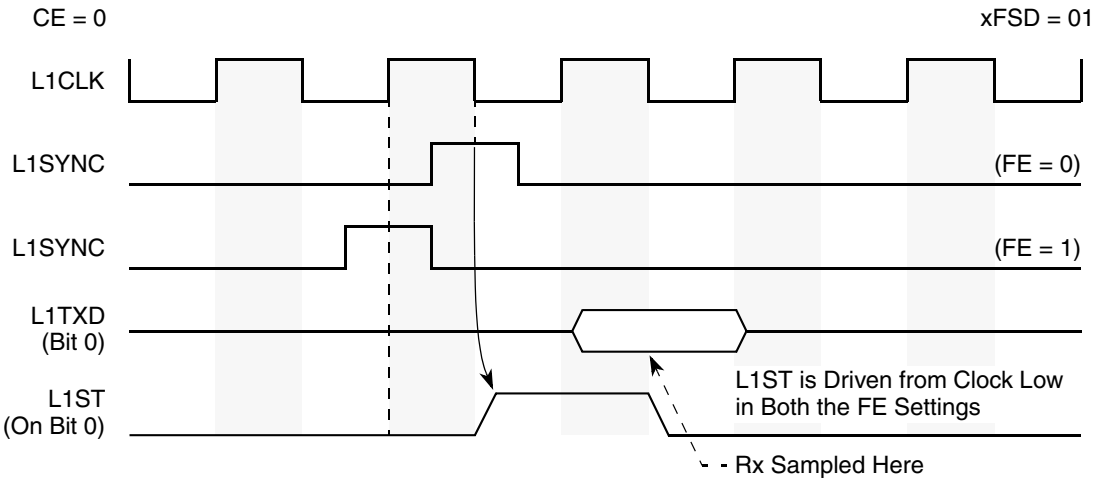


Figure 23-15. FE Effect When CE = 0 and *x*FSD = 01

Figure 23-16 shows the effects of changing FE when CE = 1 with no frame sync delay.

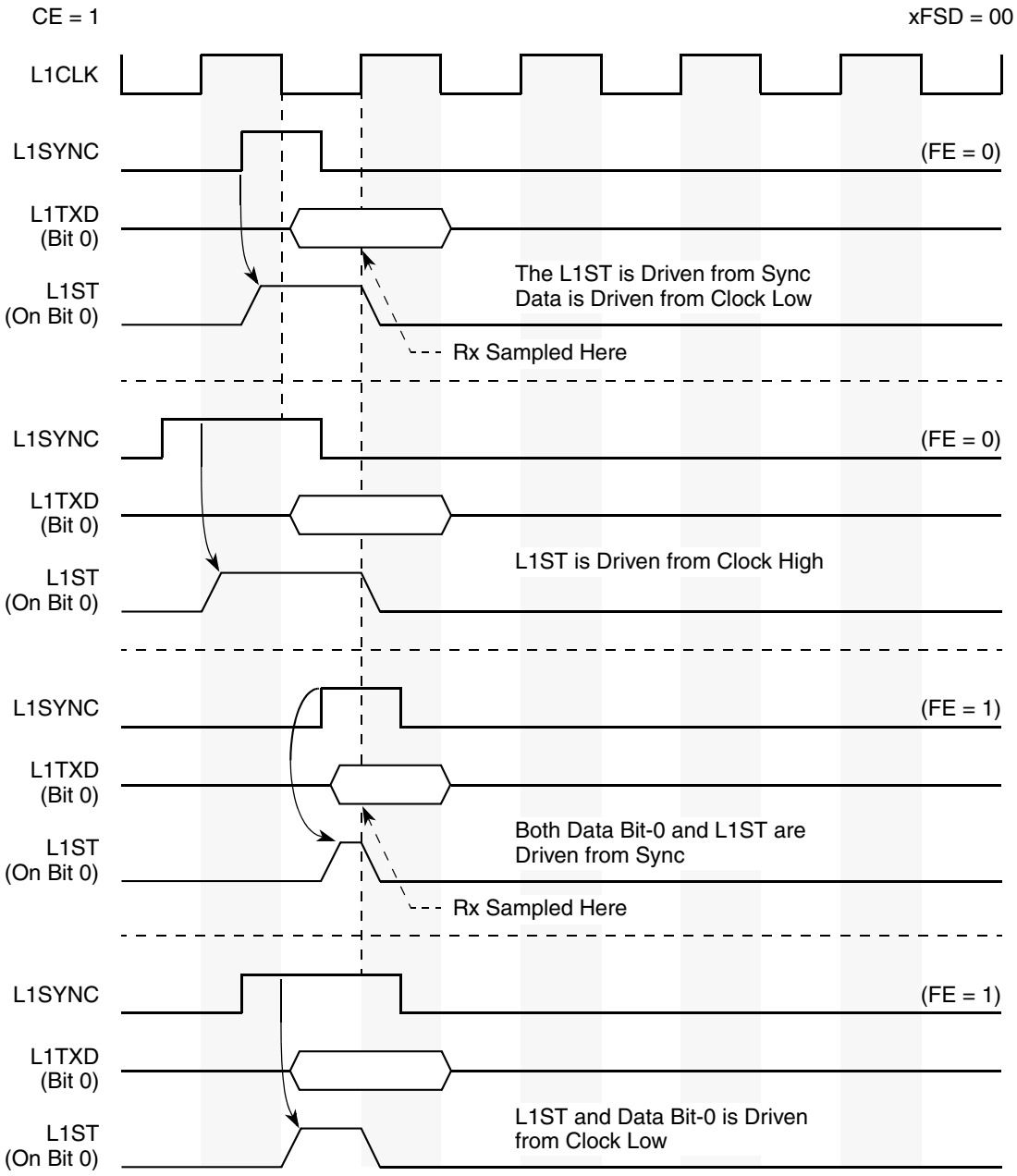


Figure 23-16. Falling Edge (FE) Effect When CE = 1 and *x*FSD = 00

Figure 23-17 shows the effects of changing FE when CE = 0 with no frame sync delay.

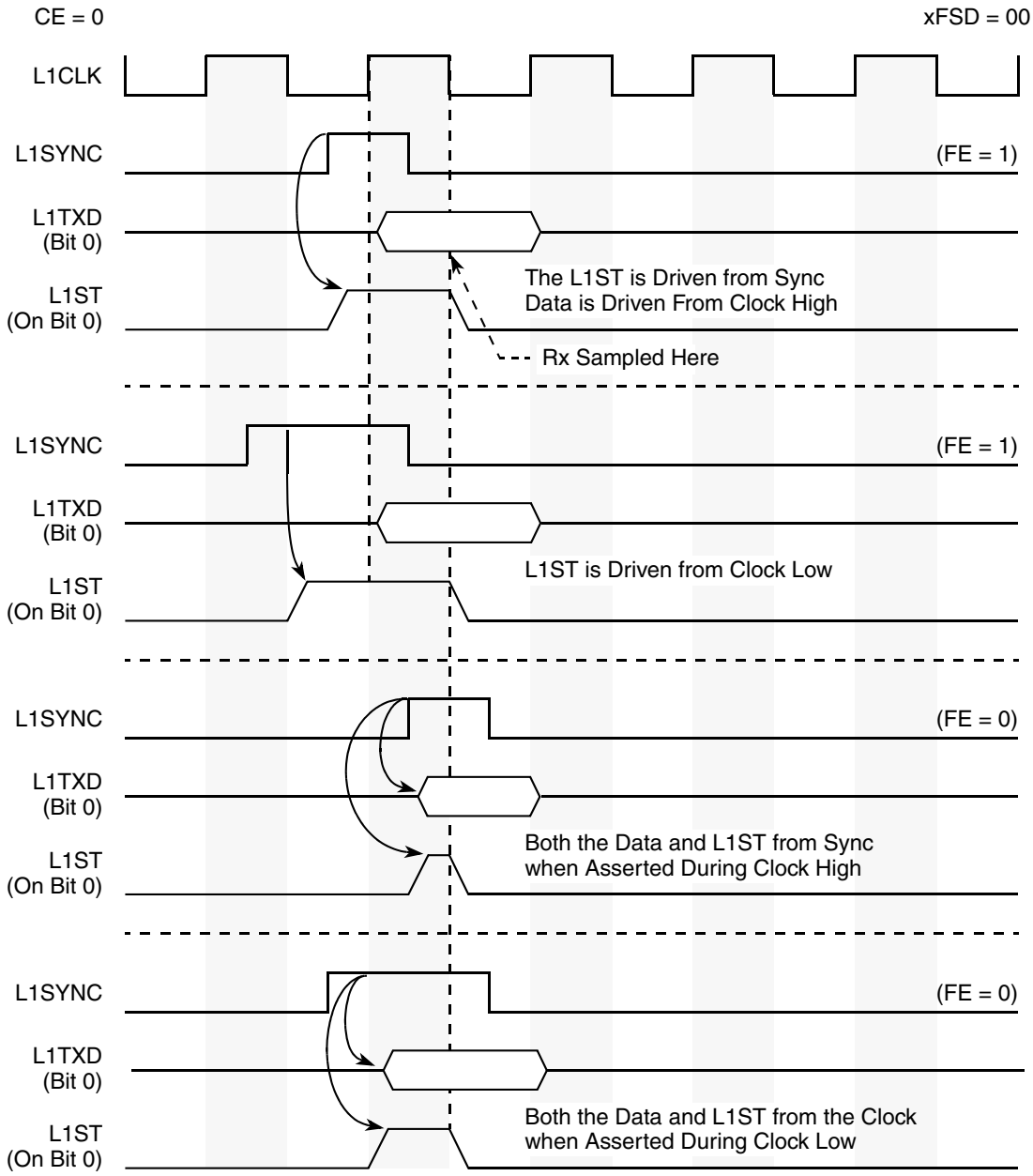


Figure 23-17. Falling Edge (FE) Effect When CE = 0 and *x*FSD = 00

23.6.3 SI2 RAM Shadow Address Registers (SI2RSR)

The SI2 RAM shadow address registers (SI2RSR), shown in Figure 23-18, define the starting addresses of the shadow section in the SI2 RAM for each of the TDM channels.

	0	1–3	4	5–7	8	9–11	12–15
Field	—	SSADA	—	SSADB	—	SSADC	—
Reset	0000_0000_0000_0000						
R/W	R/W						
Offset	0x9_1B4E (SI2RSR)						

Figure 23-18. SI2 RAM Shadow Address Registers (SI2RSR)

Table 23-5 describes SI2RSR fields.

Table 23-5. SI2RSR Field Descriptions

Bits	Name	Description
0, 4, 8, 12–15	—	Reserved, should be cleared.
1–3, 5–7, 9–11	SSAD*x*	Starting bank address for the shadow RAM of TDM a, b, or c. Defines the starting bank address of the shadow SI2 RAM section that belongs to the corresponding TDM channel. **Note:** As noted before, the SI2 RAM contain four banks of 64 entries for receive and four banks of 64 entries for transmit. In spite of the above, the starting bank address of each TDM can be programmed by the user in a granularity of 32 entries, but the user cannot put two different TDMs on the same bank. The user can put the shadow RAM section of the same TDM on the same bank. The last entry of a certain TDM frame is determined by the LST bit in the SI2 RAM entry. The user must set this bit within the entries of SI2 RAM shadow blocks for every TDM used. That means before the starting address of the next TDM.

23.6.4 SI Command Register (SI2CMDR)

The SI command registers (SI2CMDR), shown in Figure 23-19, allow the user to dynamically program the SI2 RAM. When the user sets bits in the SI2CMDR, the SI2 switches to the shadow SI2 RAM at the end of the current-route RAM programming frame. For more information about dynamic programming, see Section 23.5.5, “Static and Dynamic Routing.”

	0	1	2	3	4	5	6	7
Field	CSRRA	CSRTA	CSRRB	CSRTB	CSRRC	CSRTC	—	—
Reset	0000_0000							
R/W	R/W							
Offset	0x9_1B4A (SI2CMDR)							

Figure 23-19. SI Command Register (SI2CMDR)

Table 23-6 describes SI2CMDR fields.

Table 23-6. SI2CMDR Field Descriptions

Bits	Name	Description
0, 2, 4	CSRRx	Change shadow RAM for TDM a, b, or c receiver. Set CSRRx causes the SI receiver to replace the current route RAM with the shadow RAM. Set by the user and cleared by the SI. 0 The receiver shadow RAM is not valid. The user can write into the shadow RAM to program a new routing. 1 The receiver shadow RAM is valid. The SI exchanges between the RAMs and take the new receive routing from the receiver shadow RAM. Cleared as soon as the switch has completed.
1, 3, 5	CSRTx	Change shadow RAM for TDM a, b, or c transmitter. Set CSRTx causes the SI transmitter to replace the current route RAM with the shadow RAM. Set by the user and cleared by the SI. 0 The transmitter shadow RAM is not valid. The user can write into the shadow RAM to program a new routing. 1 The transmitter shadow RAM is valid. The SI exchanges between the RAMs and take the new transmitter routing from the receiver shadow RAM. Cleared as soon as the switch has completed.
6–7	—	Reserved, should be cleared.

23.6.5 SI Status Registers (SI2STR)

The SI status register (SI2STR), shown in Figure 23-20, identifies the current-route RAM. SI2STR values are valid only when the corresponding SI2CMDR bit = 0.

	0	1	2	3	4	5	6	7
Field	CRORA	CROTA	CRORB	CROTB	CRORC	CROTC	—	—
Reset	0000_0000							
R/W	R							
Offset	0x9_1B4C (SI2STR)							

Figure 23-20. SI Status Registers (SI2STR)

Table 23-7 describes SI2STR fields.

Table 23-7. SI2STR Field Descriptions

Bits	Name	Description
0, 2, 4	CRORx	Current-route original receiver. Determines whether the current-route receiver RAM is the original or the shadow. 0 The current-route receiver RAM is the lower address area. 1 The current-route receiver RAM is the upper address area.
1, 3, 5	CROTx	Current-route original transmitter. Determines whether the current-route transmitter RAM is the original or the shadow. 0 The current-route transmitter RAM is the lower address area. 1 The current-route transmitter RAM is the upper address area.
6–7	—	Reserved, should be cleared.

23.7 MPC8555E Serial Interface IDL Interface Support

The IDL interface is a full-duplex ISDN interface used to connect a physical layer device to the MPC8555E. The MPC8555E supports both the basic and primary rate of the IDL bus. In the basic rate of IDL, data on three channels (B1, B2, and D) is transferred in a 20-bit frame, providing a full-duplex bandwidth of 160 Kbps. The MPC8555E is an IDL slave device that is clocked by the IDL bus master (physical layer device) and has separate receive and transmit sections. The MPC8555E has three TDMs, and it can support only three independent IDL buses (limited by the number of TDMs) using separate clocks and sync pulses. Figure 23-21 shows an application with two IDL buses.

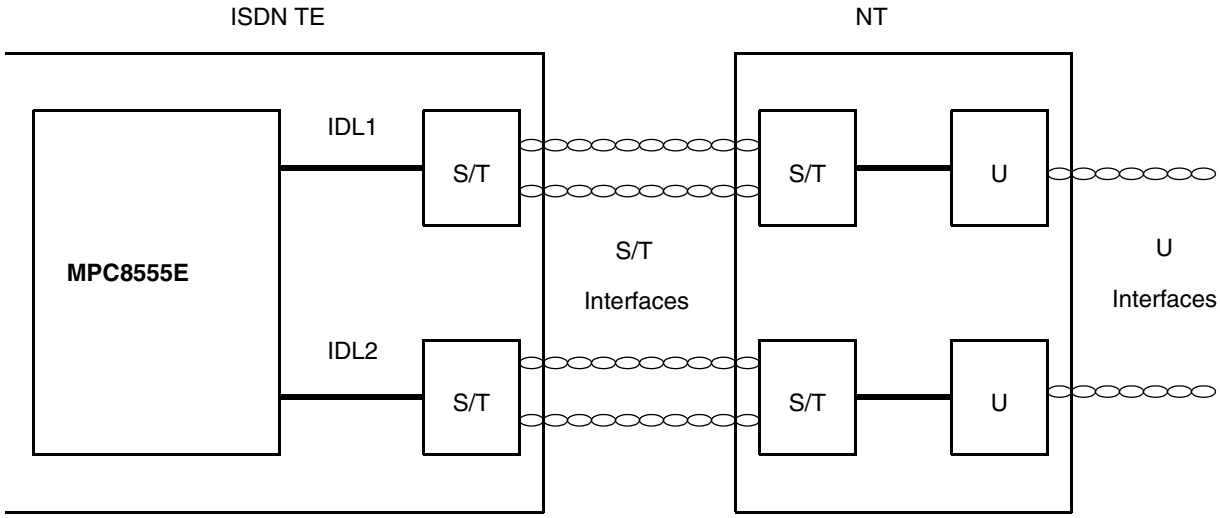


Figure 23-21. Dual IDL Bus Application Example

23.7.1 IDL Interface Example

An example of the IDL application is the ISDN terminal adaptor shown in Figure 23-22. In such an application, the IDL interface is used to connect the 2B+D channels between the MPC8555E, CODEC, and S/T transceiver. One of the MPC8555E SCCs is configured to HDLC mode to handle the D channel; another MPC8555E SCC is used to rate adapt the terminal data stream over the first B channel. That SCC is configured for HDLC mode if V.120 rate adoption is required. The second B channel is then routed to the CODEC as a digital voice channel, if preferred. The SPI is used to send initialization commands and periodically check status from the S/T transceiver. The SMC connected to the terminal is configured for UART.

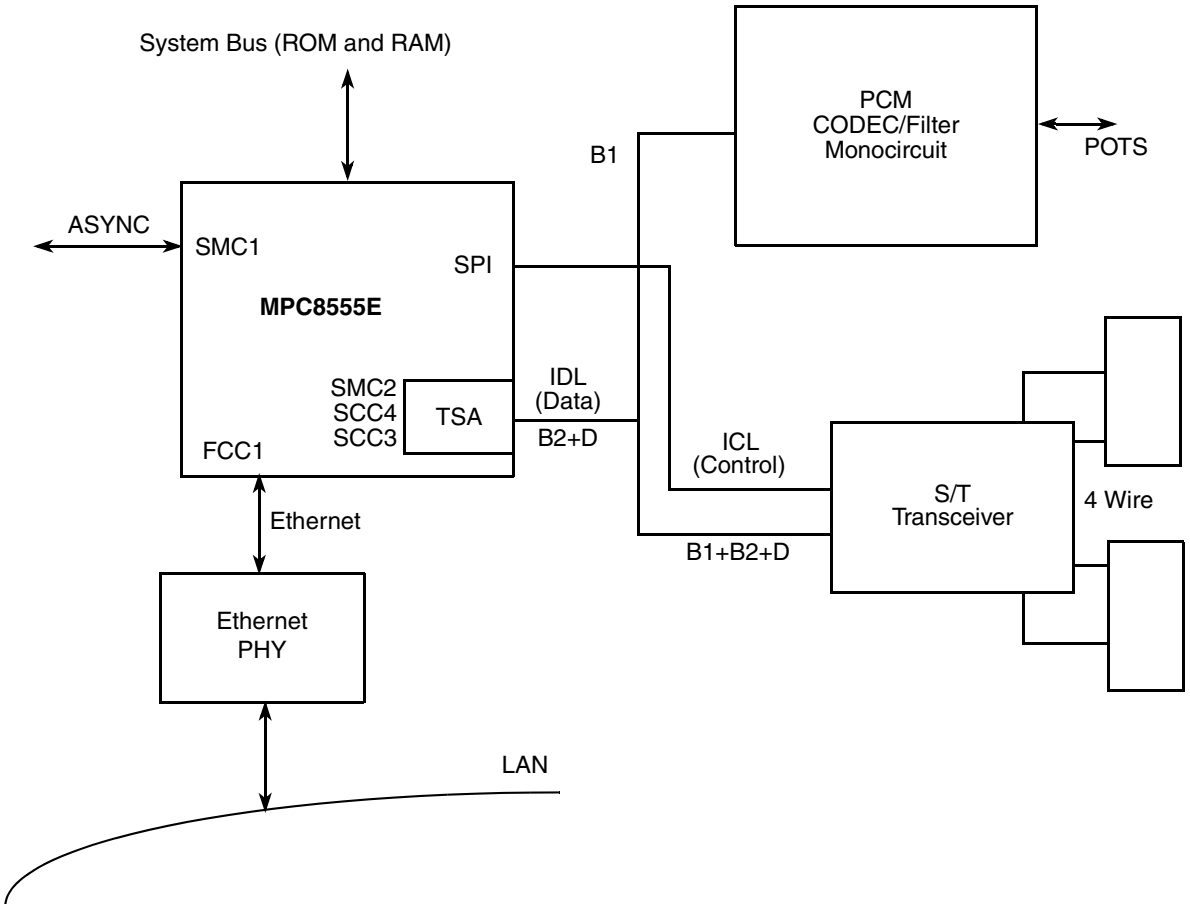


Figure 23-22. IDL Terminal Adaptor

The MPC8555E can identify and support each IDL channel or can output strobe lines for interfacing devices that do not support the IDL bus. The IDL signals for each transmit and receive channel are described in Table 23-8.

Table 23-8. IDL Signal Descriptions

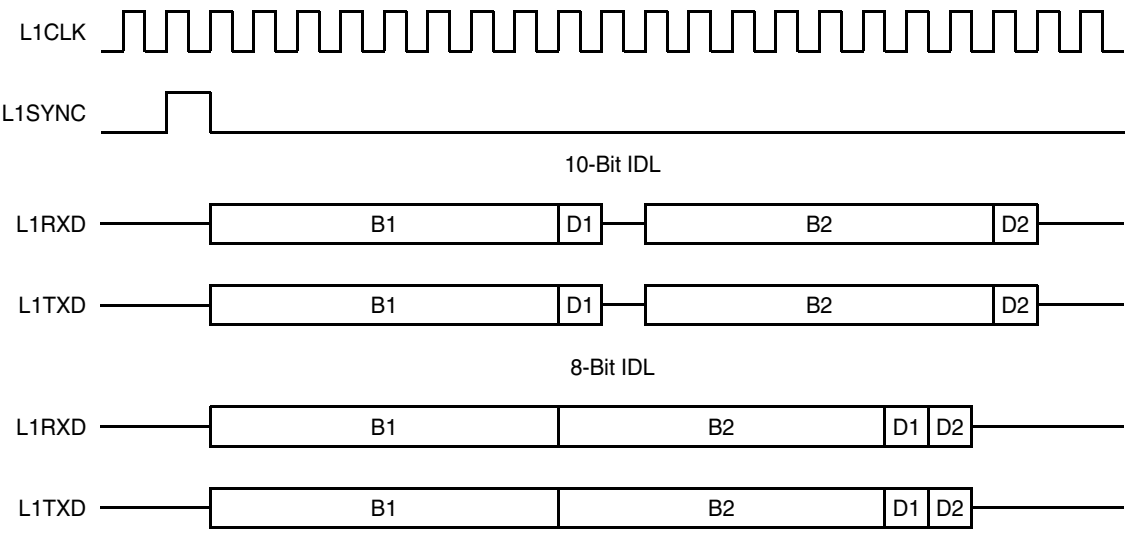
Signal	Description
L1RCLK*x*	IDL clock; input to the MPC8555E
L1RSYNC*x*	IDL sync signal; input to the MPC8555E. This signal indicates that the clock periods following the pulse designate the IDL frame.
L1RXD*x*	IDL receive data; input to the MPC8555E. Valid only for the bits supported by the IDL; ignored for any other signals present.
L1TXD*x*	IDL transmit data; output from the MPC8555E. Valid only for the bits that are supported by the IDL; otherwise, three-stated.
$\overline{\text{L1RQ}x}$	IDL request permission to transmit on the D channel; output from the MPC8555E on the $\overline{\text{L1RQ}x}$ pin.
L1GR*x*	IDL grant permission to transmit on the D Channel; input to the MPC8555E on the L1TSYNCx pin.

Note: *x* = a, b, or c for TDMa and TDMb, and TDMc.

The basic rate IDL bus has the three following channels:

- B1 is a 64-Kbps bearer channel
- B2 is a 64-Kbps bearer channel
- D is a 16-Kbps signaling channel

There are two definitions of the IDL bus frame structure—8 and 10 bits. The only difference between them is the channel order within the frame. See Figure 23-23.



Notes:
1. Clocks are not to scale.
2. $\overline{\text{L1RQ}x}$ and L1GR*x* are not shown.

Figure 23-23. IDL Bus Signals

NOTE

Previous versions of Freescale IDL-defined bit functions called auxiliary (A) and maintenance (M) were removed from the IDL definition when it was concluded that the IDL control channel would be out-of-band. These functions were defined as a subset of the Freescale SPI format called serial control port (SCP). To implement the A and M bits as originally defined, program the TSA to access these bits and route them transparently to an SCC or SMC. Use the SPI to perform out-of-band signaling.

The MPC8555E supports all channels of the IDL bus in the basic rate. Each bit in the IDL frame can be routed to any SCC and SMC or can assert a strobe output for supporting an external device. The MPC8555E supports the request-grant method for contention detection on the D channel of the IDL basic rate and when the MPC8555E has data to transmit on the D channel, it asserts $\overline{\text{L1RQ}x}$. The physical layer device monitors the physical layer bus for activity on the D channel and indicates that the channel is free by asserting L1GR*x*. The MPC8555E samples the L1GRx signal when the IDL sync signal (L1RSYNC*x*) is asserted. If L1GRx is asserted, the MPC8555E sends the first zero of the opening flag in the first bit of the D channel. If a collision is detected on the D channel, the physical layer device negates L1GR*x*. The MPC8555E then stops sending and retransmits the frame when L1GR*x* is reasserted. This procedure is handled automatically for the first two buffers of a frame.

For the primary rate IDL, the MPC8555E supports up to four 8-bit channels in the frame, determined by the SI2 RAM programming. To support more channels, the user can route more than one channel to each SCC and the SCC treats it as one high-speed stream and store it in the same data buffers (appropriate only for transparent data). Additionally, the MPC8555E can be used to assert strobes for support of additional external IDL channels.

The IDL interface supports the CCITT I.460 recommendation for data-rate adaptation since it separately accesses each bit of the IDL bus. The current-route RAM specifies which bits are supported by the IDL interface and by which serial controller. The receiver only receives bits that are enabled by the receiver route RAM. Otherwise, the transmitter sends only bits that are enabled by the transmitter route RAM and three-states L1TXD*x*.

23.7.2 IDL Interface Programming

To program an IDL interface, first program SI2MR[GM*x*] to the IDL grant mode for that channel. If the receive and transmit sections interface to the same IDL bus, set SI2MR[CRT*x*] to internally connect the Rx clock and sync signals to the transmit section. Then, program the SI2 RAM used for the IDL channels to the preferred routing. See Section 23.5.4, “SI2 RAM Programming Example.”

Define the IDL frame structure by programming SI2MR[*x*FSD*x*] to have a 1-bit delay from frame sync to data, SI2MR[FE*x*] to sample on the falling edge, and SI2MR[CE*x*] to transmit on the rising edge of the clock. Program the parallel I/O open-drain register so that L1TXD*x* is three-stated when inactive; see Section 45.2.1, “Port Open-Drain Registers (PODRx).” To support the D channel, program the appropriate CMXSCR[GR*x*] bit, as described in Section 24.4.4, “CMX SCC Clock Route Register (CMXSCR),” and program the SI2 RAM entry to route data to the chosen serial controller. The two definitions of IDL, 8 or 10 bits, are implemented by simply modifying the SI2 RAM programming. In both cases, L1GR*x* is sampled while L1TSYNC*x* is asserted and transferred to the D-channel SCC as a grant indication.

For example, based on the same 10-bit format as in Section 23.5.4, “SI2 RAM Programming Example,” implement an IDL bus using SCC1, SCC3, and SMC1 connected to TDMa2 as follows:

1. Program both the Tx and Rx sections of the SI2 RAM as in Table 23-9.

Table 23-9. SI2 RAM Entries for an IDL Interface

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0000	0011	000	1	0	8-bit SCC3
1	0	0000	0001	000	0	0	1-bit SCC1
2	0	0000	0000	000	0	0	1-bit no support
3	0	0000	0101	011	0	0	4-bit SMC1
4	0	0000	0101	011	0	0	4-bit SMC1
5	0	1000	0001	000	0	1	1-bit SCC1 strobe1

2. CMXSI2CR = 0x00. TDMA receive clock is CLK13.

3. CMXSMR = 0x80. SMC1 is connected to the TSA.
4. CMXSCR = 0xC000_4000. SCC1 and SCC3 are connected to the TSA. SCC1 supports the grant mechanism because it handles the D channel.
5. SI2AMR = 0x0145. TDMA grant mode is used with 1-bit frame sync delay in Tx and Rx and common receive-transmit mode.
6. Set PPARD[20–22], PPARC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
7. Set PSORD[20–22], PSORC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
8. Clear PDIRD[22]. Configures L1TXDA(inout).
9. Set PODRD[22]. Configures L1TXDA to an open-drain output.
10. Set PPARC[19]. Configures L1RCLKA.
11. Clear PDIRC[19]. Configures L1RCLKA.
12. Clear PSORC[19]. Configures L1RCLKA.
13. Set PPARC[1]. Configures $\overline{\text{L1RQa}}$.
14. Set PSORC[1]. Configures $\overline{\text{L1RQa}}$.
15. Set PDIRC[1]. Configures $\overline{\text{L1RQa}}$.
16. Set PPARC[8]. Configures L1ST1.
17. Set PSORC[8]. Configures L1ST1.
18. Set PDIRC[8]. Configures L1ST1.
19. SI2CMDR is not used.
20. SI2STR does not need to be read.
21. Configure the SCC1 for HDLC operation (to handle the LAPD protocol of the D channel), and configure SCC3 and SMC1 as preferred.
22. SI2GMR = 0x01. Enable TDM A (one static TDM).
23. Enable SCC1, SCC3, and SMC1.

23.8 Serial Interface GCI Support

The MPC8555E fully supports the normal mode of the GCI, also known as the ISDN-oriented modular revision 2.2 (IOM-2), and the SCIT. The MPC8555E also supports the D-channel access control in S/T interface terminals using the command/indication (C/I) channel

The GCI bus consists of four lines—two data lines, a clock, and a frame synchronization line. Usually, an 8-kHz frame structure defines the various channels within the 256-Kbps data rate. The MPC8555E supports two (limited by the number of SMCs and TDMs) independent GCI buses, each with independent receive and transmit sections. The interface can also be used in a multiplexed frame structure on which up to eight physical layer devices multiplex their GCI channels. In this mode, the data rate would be 2,048 Kbps.

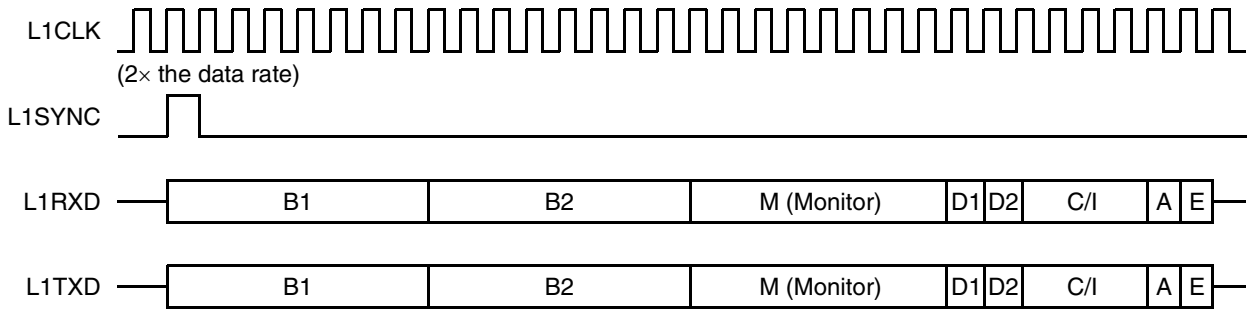
In the GCI bus, the clock rate is twice the data rate. The SI divides the input clock by two to produce the data clock. The MPC8555E also has data strobe lines and the 1× data rate clock L1CLKOx output pins. These signals are used for interfacing devices to GCI that do not support the GCI bus. Table 23-10 describes GCI signals for each transmit and receive channel.

Table 23-10. GCI Signals

Signal	Description
L1RSYNC*x*	Used as a GCI sync signal; input to the MPC8555E. This signal indicates that the clock periods following the pulse designate the GCI frame.
L1RCLK*x*	Used as a GCI clock; input to the MPC8555E. The L1RCLK*x* signal frequency is twice the data clock.
L1RXD*x*	Used as a GCI receive data; input to the MPC8555E.
L1TXD*x*	Used as a GCI transmit data; open-drain output. Valid only for the bits that are supported by the IDL; otherwise, three-stated.
L1CLKO*x*	Optional signal; output from the MPC8555E. This 1× clock output is used to clock devices that do not interface directly to the GCI. If the double-speed clock is used, (DSC*x* bit is set in the SI2MR), this output is the L1RCLK*x* divided by 2; otherwise, it is simply a 1× output of the L1RCLK*x* signal.

Note: *x* = a, b, and c for TDMa and TDMb and TDMc.

The GCI bus signals are shown in Figure 23-24.



Notes: Clock is not to scale.
L1CLKO is not shown.

Figure 23-24. GCI Bus Signals

In addition to the 144-Kbps ISDN 2B+D channels, the GCI provides five channels for maintenance and control functions:

- B1 is a 64-Kbps bearer channel
- B2 is a 64-Kbps bearer channel
- M is a 64-Kbps monitor channel
- D is a 16-Kbps signaling channel
- C/I is a 48-Kbps C/I channel (includes A and E bits)

The M channel is used to transfer data between layer 1 devices and the control unit (the CPU); the C/I channel is used to control activation/deactivation procedures or to switch test loops by the control unit. The M and C/I channels of the GCI bus should be routed to SMC1 or SMC2, which have modes to support the channel protocols. The MPC8555E can support any channel of the GCI bus in the primary rate by modifying SI2 RAM programming.

The GCI supports the CCITT I.460 recommendation as a method for data rate adaptation since it can access each bit of the GCI separately. The current-route RAM specifies which bits are supported by the

interface and which serial controller support them. The receiver only receives the bits that are enabled by the SI2 RAM and the transmitter only transmits the bits that are enabled by the SI2 RAM and does not drive L1TXDx. Otherwise, L1TXDx is an open-drain output and should be pulled high externally.

The MPC8555E supports contention detection on the D channel of the SCIT bus. When the MPC8555E has data to transmit on the D channel, it checks a SCIT bus bit that is marked with a special route code (usually, bit 4 of C/I channel 2). The physical layer device monitors the physical layer bus for activity on the D channel and indicates on this bit that the channel is free. If a collision is detected on the D channel, the physical layer device sets bit 4 of C/I channel 2 to logic high. The MPC8555E then aborts its transmission and retransmits the frame when this bit is set again. This procedure is automatically handled for the first two buffers of a frame.

23.8.1 SI GCI Activation/Deactivation Procedure

In the deactivated state, the clock pulse is disabled and the data line is at a logic one. The layer 1 device activates the MPC8555E by enabling the clock pulses and by an indication in the channel 0 C/I channel. The MPC8555E reports to the core (through a maskable interrupt) that a valid indication is in the SMC RxBD.

When the core activates the line, the data output of L1TXDn is programmed to zero by setting SI2GMR[STZx]. Code 0 (command timing TIM) is transmitted on channel 0 C/I channel to the layer 1 device until STZx is reset. The physical layer device resumes the clock pulses and gives an indication in the channel 0 C/I channel. The core should reset STZx to enable data output.

23.8.2 Serial Interface GCI Programming

The following sections describe serial interface GCI programming.

23.8.2.1 Normal Mode GCI Programming

The user can program and configure the channels used for the GCI bus interface. First, the SI2MR register to the GCI/SCIT mode for that channel must be programmed, using the DSC*x*, FE*x*, CE*x*, and RFSD*x* bits. This mode defines the sync pulse to GCI sync for framing and data clock as one-half the input clock rate. The user can program more than one channel to interface to the GCI bus. Also, if the receive and transmit section are used for interfacing the same GCI bus, the user internally connects the receive clock and sync signals to the SI2 RAM transmit section, using the CRT*x* bits. The user should then define the GCI frame routing and strobe select using the SI2 RAM.

When the receive and transmit section uses the same clock and sync signals, these sections should be programmed to the same configuration. Also, the L1TXD*x* pin in the I/O register should be programmed to be an open-drain output. To support the monitor and the C/I channels in GCI, those channels should be routed to one of the SMCs. To support the D channel when there is no possibility of collision, the user should clear the SI2MR[GR*x*] bit corresponding to the SCC that supports the D channel.

23.8.2.2 SCIT Programming

For interfacing the GCI/SCIT bus, SI2MR must be programmed to the GCI/SCIT mode. The SI2 RAM is programmed to support a 96-bit frame length and the frame sync is programmed to the GCI sync pulse. Generally, the SCIT bus supports the D channel access collision mechanism. For this purpose, the user should program the CRT*x* bits so the receive and transmit sections use the same clock and sync signals and program the GR*x* bits to transfer the D channel grant to the SCC that supports this channel. The received (grant) bit should be marked by programming the channel select bits of the SI2 RAM to 0b0111 for an internal assertion of a strobe on this bit. This bit is sampled by the SI and transferred to the D-channel SCC as the grant. The bit is generally bit 4 of the C/I in channel 2 of the GCI, but any other bit can be selected using the SI2 RAM.

For example, assuming that SCC1 is connected to the D channel, SCC3 to the B1 channel, and SMC2 to the B2 channel, SMC1 is used to handle the C/I channels, and the D-channel grant is on bit 4 of the C/I on SCIT channel 2, the initialization sequence is as follows:

1. Program both the Tx and Rx sections of the SI2 RAM as in Table 23-11 beginning at addresses 0 and 1024, respectively.

Table 23-11. SI2 RAM Entries for a GCI Interface (SCIT Mode)

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0000	0011	000	1	0	8 bits SCC3
1	0	0000	0110	000	1	0	8 bits SMC2
2	0	0000	0101	000	1	0	8 bits SMC1
3	0	0000	0001	001	0	0	2 bits SCC1
4	0	0000	0101	101	0	0	6 bits SMC1
5	0	0000	0000	110	1	0	Skip 7 bytes
6	0	0000	0000	001	0	0	Skip 2 bits
7	0	0000	0111	000	0	1	D grant bit

2. SI2AMR = 0x00c0. TDMa is used in double speed clock and common Rx/Tx modes. SCIT mode is used in this example.

NOTE

If SCIT mode is not used, delete the last three entries of the SI2 RAM, divide one entry into two, and set the LST bit in the new last entry.

3. CMXSMR = 0x88. SMC1 and SMC2 are connected to the TSA.
4. CMXSCR = 0xC000_4000. SCC3 and SCC1 are connected to the TSA. SCC1 supports the grant mechanism since it is on the D channel.
5. CMXSI2CR = 0x00. TDMa uses CLK13.
6. Set PPARD[20–22], PPARC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
7. Set PSORD[20–22], PSORC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.

8. Clear PDIRD[22]. Configures L1TXDA(inout).
9. Set PODRD[22]. Configures L1TXDA to an open-drain output.
10. Set PPARC[19]. Configures L1RCLKA.
11. Clear PDIRC[19]. Configures L1RCLKA.
12. Clear PSORC[19]. Configures L1RCLKA.
13. Set PPARC[1]. Configures $\overline{\text{L1RQa}}$.
14. Set PSORC[1]. Configures $\overline{\text{L1RQa}}$.
15. Set PDIRC[1]. Configures $\overline{\text{L1RQa}}$.
16. If the 1× GCI data clock is required, set PPARC bit 0 and PDIRC bit 0 and set PSORC 0, which configures L1CLKOa as an output.
17. Configure SCC1 for HDLC operation (to handle the LAPD protocol of the D channel). Configure SMC1 for SCIT operation and configure SCC3 and SMC2 as preferred.
18. SI2GMR = 0x11. Enable TDMa (one static TDM), STZ for TDMa.
19. SI2CMDR is not used.
20. SI2STR does not need to be read.
21. Enable SCC1, SCC3, SMC1, and SMC2.

Chapter 24
CPM Multiplexing

The CPM multiplexing logic (CMX) connects the physical layer—UTOPIA, MII/RMII, modem lines, TDM lines, and proprietary serial lines to the FCCs, SCCs, SMCs, and USB. The CMX features the following two modes:

- In NMSI mode, the CMX allows all serial devices to be connected to their own set of individual pins. Each serial device that connects to the external world in this way is said to connect to a nonmultiplexed serial interface (NMSI). In the NMSI configuration, the CMX provides a flexible clocking assignment for each FCC, SCC, SMC, and USB from a bank of external clock pins and/or internal BRGs.
- In TDM mode, the CMX performs the connection of the serial devices to the SI for using the time-slot assigner (TSA). This allows any combination of FCCs, SCCs, and SMCs to multiplex data on any of the TDM channels. The CMX connects the serial device only to the TSA in the SI. The actual multiplexing of the TDM is made by programming the SI RAM. In TDM mode, all other pins used in NMSI mode are available for other purposes. See Chapter 23, “Serial Interface with Time-Slot Assigner.”

NOTE

The USB uses only the NMSI mode and is mutually exclusive with SCC3 in NMSI mode; it is not legal to enable both peripherals in NMSI mode at the same time. Figure 24-1 shows a block diagram of the CMX.

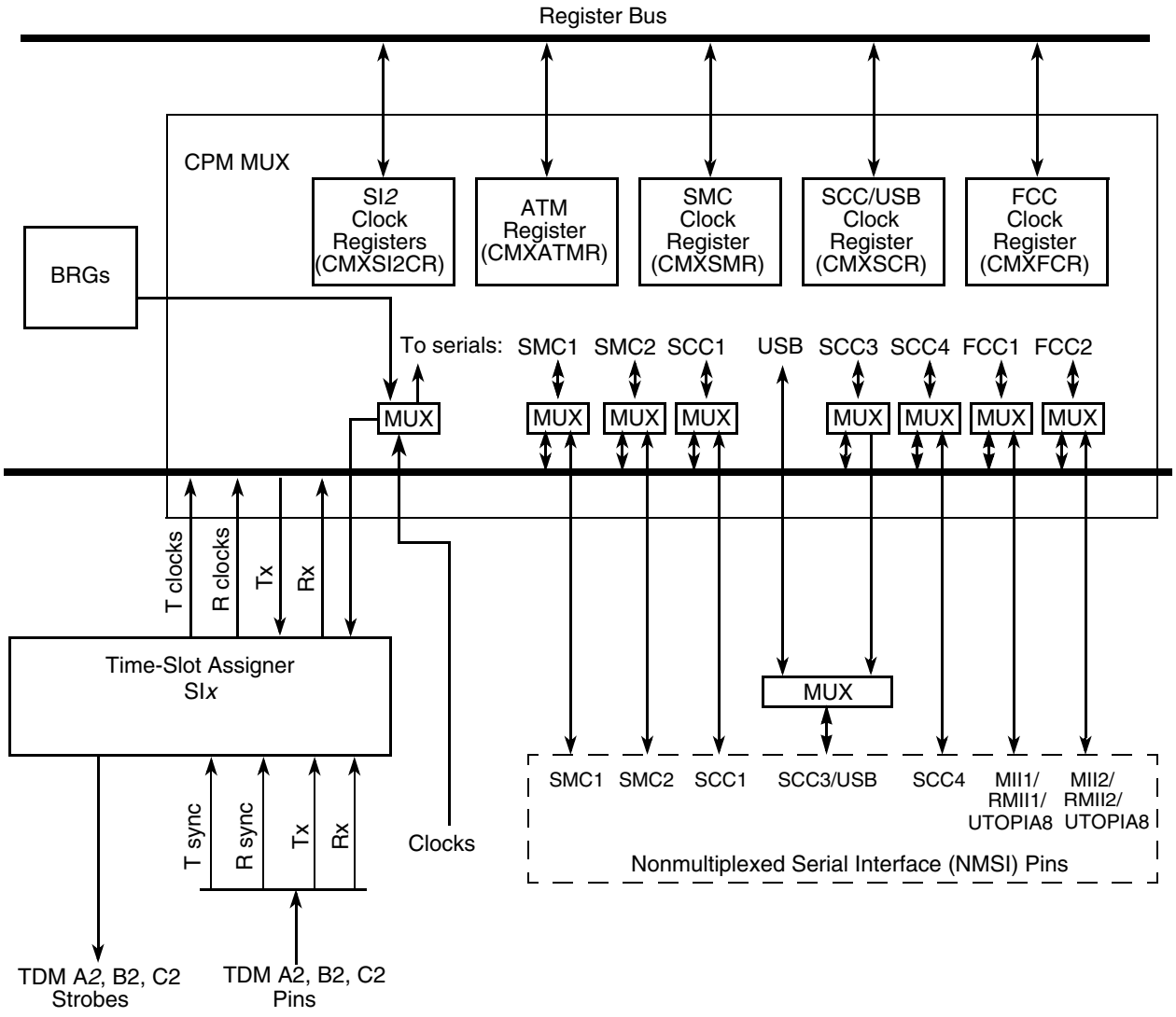


Figure 24-1. CPM Multiplexing Logic (CMX) Block Diagram

24.1 Features

The NMSI mode supports the following:

- Each FCC, SCC, and SMC can be programmed independently to work with a serial device's own set of pins in a non-multiplexed manner.
- Each FCC can be connected to its own MII/RMII (media-independent interface).
- FCC1 can also be connected to an 8-bit ATM UTOPIA level-2interface.
- FCC2 can also be connected to an 8-bit ATM UTOPIA level-2 interface.
- Each SCC can have its own set of modem control pins.
- Each SMC can have its own set of four pins.
- Each FCC, SCC, SMC, and the USB can be driven from a bank of 14 clock pins or a bank of 8 BRGs.

NOTE

In FCC2 Utopia Master or Slave mode only, Clk13, Clk14, BRG 5, BRG6, and BRG7 are available.

24.2 Enabling Connections to TSA or NMSI

Each serial device can be independently enabled to connect to the TSA or to dedicated external pins, as shown in Figure 24-2. Each FCC can be connected to a dedicated MII/RMII, the three TDMs, or to an 8-bit UTOPIA level II interface. Each SCC or SMC can be connected to the three TDMs or to its own set of pins. Once connections are made to the TSA, the exact routing decisions are made in the SI RAM.

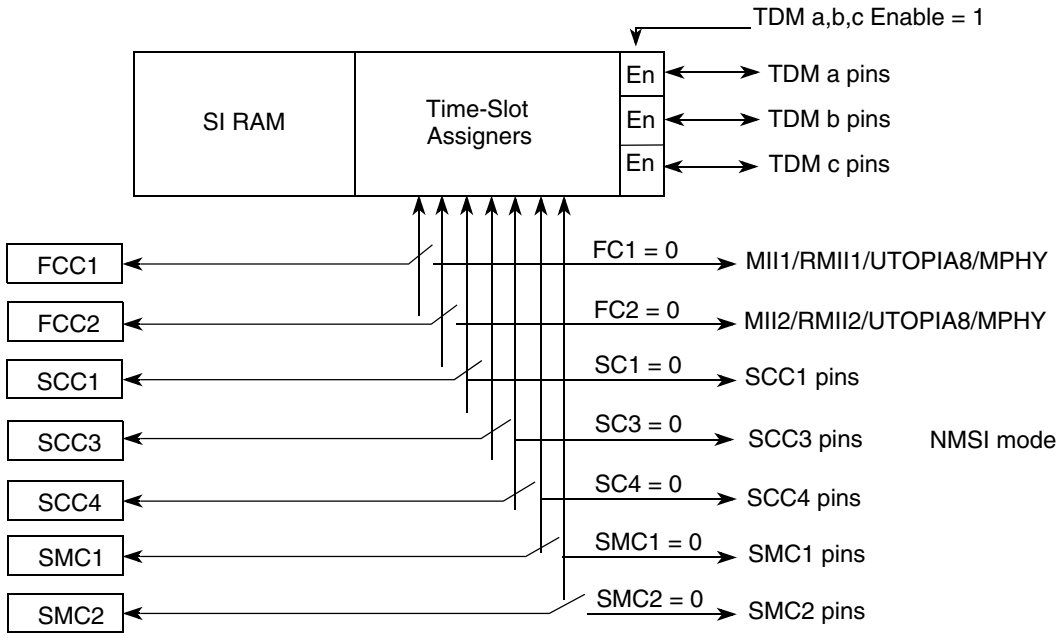


Figure 24-2. Enabling Connections to the TSA

24.3 NMSI Configuration

The CMX supports an NMSI mode for each of the FCCs, SCCs, and SMCs. Each of these serial devices is connected independently either to the NMSI or to the TSA using the clock route registers. The user should note, however, that NMSI pins are multiplexed with other functions at the parallel I/O lines. Therefore, if a combination of TDM and NMSI channels are used, consult the pinout to determine which FCC, SCC, and SMC to connect and where to connect them.

The clocks provided to the FCCs, SCCs, SMCs, and USB are derived from a bank of 8 internal BRGs and 14 external CLK pins; see Figure 24-3. There are two main advantages to the bank-of-clocks approach. First, a serial device is not forced to choose a serial device clock from a predefined pin or BRG; this allows a flexible pinout-mapping strategy. Second, a group of serial receivers and transmitters that needs the same clock rate can share the same pin. This configuration leaves additional pins for other functions and minimizes potential skew between multiple clock sources.

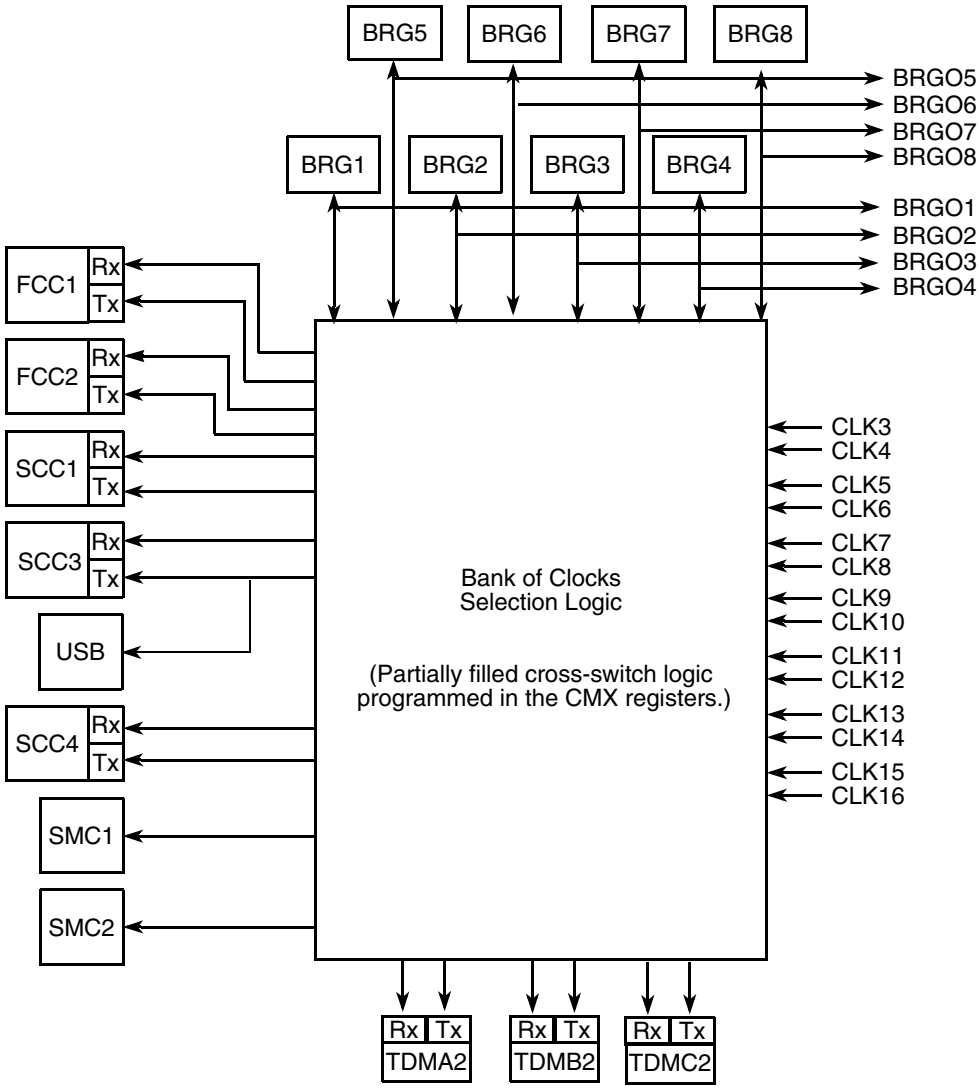


Figure 24-3. Bank of Clocks

The eight BRGs also make their clocks available to external logic, regardless of whether the BRGs are being used by a serial device. Notice that the BRG outputs are multiplexed with other functions; thus, all BRGO*x* pins may not always be available. Chapter 45, "Parallel I/O Ports," shows the function multiplexing.

There are two restrictions in the bank-of-clocks mapping:

- Only 4 of the 14 sources can be connected to any given FCC, SCC receiver or transmitter or to the USB.
- The SMC transmitter and receiver share the same clock source when connected to the NMSI.

Table 24-1 shows the clock source options for the serial controllers and TDM channels.

Table 24-1. Clock Source Options

Clock	CLK														BRG							
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
SCC1 Rx	V	V							V	V					V	V	V	V				
SCC1 Tx	V	V							V	V					V	V	V	V				
SCC3 Rx			V	V	V	V									V	V	V	V				
SCC3 Tx/USB			V	V	V	V									V	V	V	V				
SCC4 Rx			V	V	V	V									V	V	V	V				
SCC4 Tx			V	V	V	V									V	V	V	V				
FCC1 Rx							V	V	V	V									V	V	V	V
FCC1 Tx							V	V	V	V									V	V	V	V
FCC2 Rx											V	V	V	V					V	V	V	V
FCC2 Tx											V	V	V	V					V	V	V	V
TDMA2 Rx			V								V											
TDMA2 Tx				V								V										
TDMB2 Rx	V						V															
TDMB2 Tx		V						V														
TDMC2 Rx			V								V											
TDMC2 Tx				V								V										
SMC1 Rx					V		V								V						V	
SMC1 Tx					V		V								V						V	
SMC2 Rx		V											V			V						V
SMC2 Tx		V											V			V						V

NOTE

After a clock source is selected, the clock is given an internal name. For the FCCs and SCCs, the names are RCLK*x* and TCLK*x*; for SMCs, the name is simply SMCLK*x*; for USB the name is USBCLK. These internal names specify the clocks sent to the FCCs, SCCs, SMCs, or USB. The internal names for the FCCs, SCCs, or SMCs are used only in NMSI. These names do not correspond to any MPC8555E pins.

24.4 CMX Registers

The following sections describe the CMX registers.

24.4.1 CMX UTOPIA Address Register (CMXUAR)

The CMX UTOPIA address register (CMXUAR), shown in Figure 24-4, defines the connection of FCC1 and FCC2 UTOPIA multiple-PHY addresses to the 14 UTOPIA address pins of the MPC8555E; it also defines the connection of a BRG to the FCCs when an internal rate feature is used. This enables the user to implement a multiple-PHY UTOPIA master or slave on both FCC1 and FCC2 using only 14 pins. The

user chooses the number of PHYs to use with each interface and the number of address lines needed for each FCC.

	0	1	2	3	4	5–7	8–9	10–11	12–15
Field	SAD0	SAD1	SAD2	SAD3	SAD4	—	F1IRB	F2IRB	—
Reset	0000_0000_0000_0000								
R/W	R/W								
Offset	0x9_1B0E								

Figure 24-4. CMX UTOPIA Address Register (CMXUAR)

Table 24-2 describes CMXUAR fields.

Table 24-2. CMXUAR Field Descriptions

Bits	Name	Description
0–4	SAD*x*	Slave address input pin *x* connection. See Note that the address indexes are relative to FCC1; see Figure 24-5. 0 Reserved. 1 This address input pin is used by FCC1 in slave mode.
5–7	—	Reserved, should be cleared.
8–9	F1IRB	FCC1 internal rate BRG selection. Selects the BRG to be connected to FCC1 for internal rate operation. Used by the ATM controller; see Section 41.2.1.5, “Transmit External Rate and Internal Rate Modes.” 00 FCC1 internal rate clock is BRG5 01 FCC1 internal rate clock is BRG6 10 FCC1 internal rate clock is BRG7 11 FCC1 internal rate clock is BRG8
10–11	F2IRB	FCC2 internal rate BRG selection. Selects the BRG to be connected to FCC2 for internal rate operation. Used by the ATM controller; see Section 41.2.1.5, “Transmit External Rate and Internal Rate Modes.” 00 FCC2 internal rate clock is BRG5 01 FCC2 internal rate clock is BRG6 10 FCC2 internal rate clock is BRG7 11 FCC2 internal rate clock is BRG8
12–15	—	Reserved, should be cleared.

Figure 24-5 describes the interconnection between the receive external multi-PHY bus and the internal FCC1 and FCC2 receive multi-PHY addresses. The same diagram applies to the transmit multi-PHY bus using different dedicated parallel I/O pins.

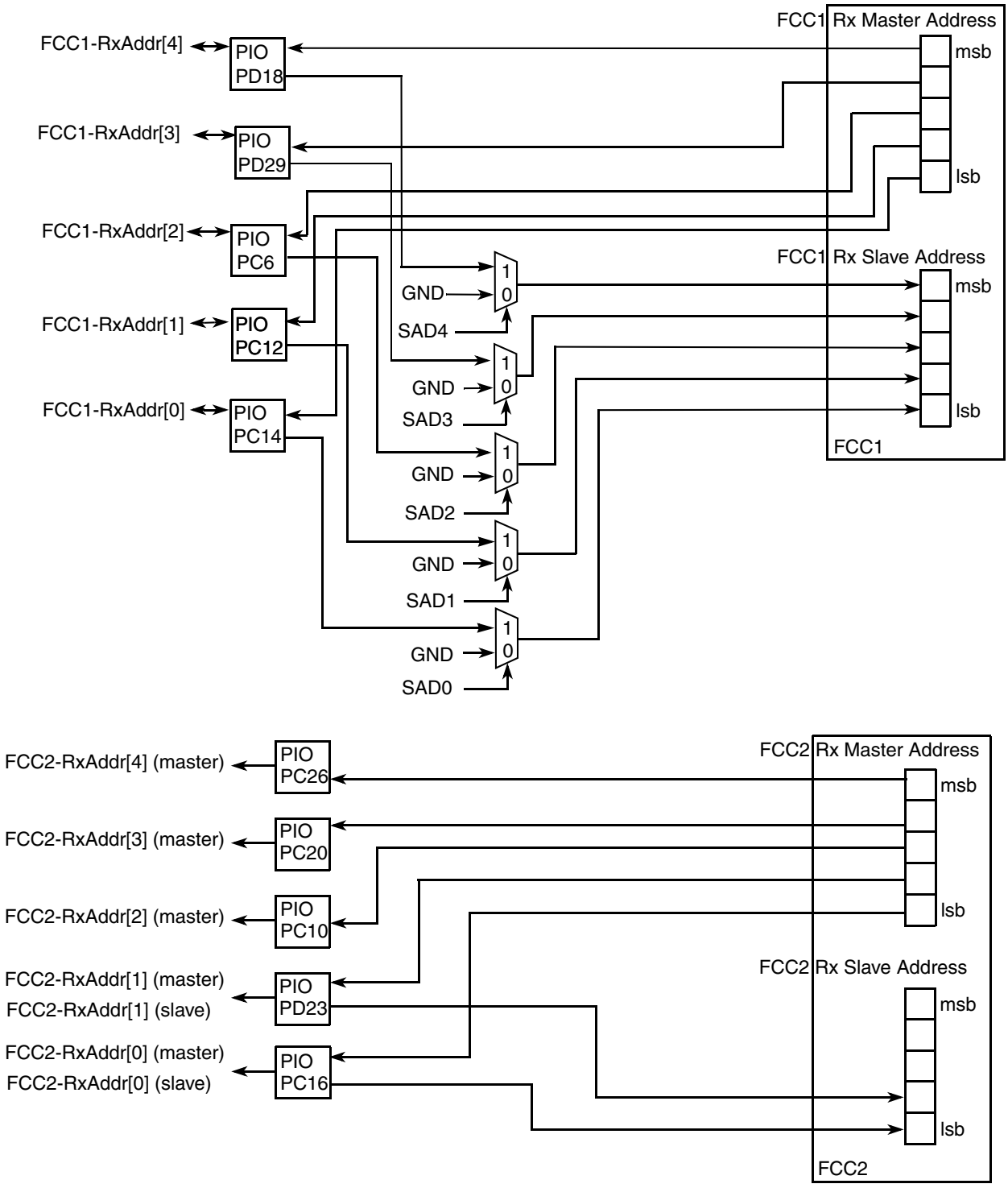


Figure 24-5. Multi-PHY Receive Address Multiplexing

24.4.2 CMX SI2 Clock Route Register (CMXSI2CR)

The CMX SI2 clock route register (CMXSI2CR), seen in Figure 24-6, defines the connection of SI2 to the clock sources that can be input from the bank of clocks.

	0	1	2	3	4	5	6	7
Field	RTA2CS	RTB2CS	RTC2CS	—	TTA2CS	TTB2CS	TTC2CS	—
Reset	0000_0000							
R/W	R/W							
Offset	0x9_1B02							

Figure 24-6. CMX SI2 Clock Route Register (CMXSI2CR)

Table 24-3 describes CMXSI2CR fields.

Table 24-3. CMXSI2CR Field Descriptions

Bits	Name	Description
0	RTA2CS	Receive TDM A2 clock source 0 TDM A2 receive clock is CLK13 1 TDM A2 receive clock is CLK5
1	RTB2CS	Receive TDM B2 clock source 0 TDM C2 receive clock is CLK3 1 TDM C2 receive clock is CLK9
2	RTC2CS	Receive TDM C2 clock source 0 TDM C2 receive clock is CLK5 1 TDM C2 receive clock is CLK13
3	—	Reserved, should be cleared.
4	TTA2CS	Transmit TDM A2 clock source 0 TDM A2 transmit clock is CLK14 1 TDM A2 transmit clock is CLK6
5	TTB2CS	Transmit TDM B2 clock source 0 TDM B2 transmit clock is CLK4 1 TDM B2 transmit clock is CLK10
6	TTC2CS	Transmit TDM C2 clock source 0 TDM C2 transmit clock is CLK6 1 TDM C2 transmit clock is CLK14
7	—	Reserved, should be cleared.

24.4.3 CMX FCC Clock Route Register (CMXFCR)

The CMX FCC clock route register (CMXFCR), shown in Figure 24-7, defines the connection of the FCCs to the TSA and to the clock sources from the bank of clocks.

	0	1	2–4	5–7	8	9	10–12	13–15
Field	—	FC1	RF1CS	TF1CS	—	FC2	RF2CS	TF2CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	0x9_1B04							

	16–31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Offset	0x9_1B06

Figure 24-7. CMX FCC Clock Route Register (CMXFCR)

Table 24-4 describes CMXFCR fields.

Table 24-4. CMXFCR Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1	FC1	Defines the FCC1 connection 0 FCC1 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC1 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
2–4	RF1CS	Receive FCC1 clock source (NMSI mode). Ignored if FCC1 is connected to the TSA (FC1 = 1). 000 FCC1 receive clock is BRG5 001 FCC1 receive clock is BRG6 010 FCC1 receive clock is BRG7 011 FCC1 receive clock is BRG8 100 FCC1 receive clock is CLK9 101 FCC1 receive clock is CLK10 110 FCC1 receive clock is CLK11 111 FCC1 receive clock is CLK12
5–7	TF1CS	Transmit FCC1 clock source (NMSI mode). Ignored if FCC1 is connected to the TSA (FC1 = 1). 000 FCC1 transmit clock is BRG5 001 FCC1 transmit clock is BRG6 010 FCC1 transmit clock is BRG7 011 FCC1 transmit clock is BRG8 100 FCC1 transmit clock is CLK9 101 FCC1 transmit clock is CLK10 110 FCC1 transmit clock is CLK11 111 FCC1 transmit clock is CLK12
8	—	Reserved, should be cleared
9	FC2	Defines the FCC2 connection 0 FCC2 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC2 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.

Table 24-4. CMXFCR Field Descriptions (continued)

Bits	Name	Description
10–12	RF2CS	Receive FCC2 clock source (NMSI mode). Ignored if FCC2 is connected to the TSA (FC2 = 1) 000 FCC2 receive clock is BRG5 001 FCC2 receive clock is BRG6 010 FCC2 receive clock is BRG7 011 FCC2 receive clock is BRG8 100 FCC2 receive clock is CLK13 101 FCC2 receive clock is CLK14 110 FCC2 receive clock is CLK15 111 FCC2 receive clock is CLK16
13–15	TF2CS	Transmit FCC2 clock source (NMSI mode). Ignored if FCC2 is connected to the TSA (FC2 = 1) 000 FCC2 transmit clock is BRG5 001 FCC2 transmit clock is BRG6 010 FCC2 transmit clock is BRG7 011 FCC2 transmit clock is BRG8 100 FCC2 transmit clock is CLK13 101 FCC2 transmit clock is CLK14 110 FCC2 transmit clock is CLK15 111 FCC2 transmit clock is CLK16
16–31	—	Reserved, should be cleared

24.4.4 CMX SCC Clock Route Register (CMXSCR)

The CMX SCC clock route register (CMXSCR), seen in Figure 24-8, defines the connection of the SCCs to the TSA and to the clock sources from the bank of clocks. This register also enables the use of the external grant pin.

	0	1	2 4	5 7	8 15
Field	GR1	SC1	RS1CS	TS1CS	—
Reset	0000_0000_0000_0000				
R/W	R/W				
Offset	0x9_1B08				

	16	17	18 20	21 23	24	25	26 28	29 31
Field	GR3	SC3	RS3CS	TS3CS	GR4	SC4	RS4CS	TS4CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	0x9_1B0A							

Figure 24-8. CMX SCC Clock Route Register (CMXSCR)

Table 24-5 describes CMXSCR fields.

Table 24-5. CMXSCR Field Descriptions

Bits	Name	Description
0	GR1	Grant support of SCC1 0 SCC1 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC1 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
1	SC1	SCC1 connection 0 SCC1 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC1 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.
2–4	RS1CS	Receive SCC1 clock source (NMSI mode). Ignored if SCC1 is connected to the TSA (SC1 = 1) 000 SCC1 receive clock is BRG1 001 SCC1 receive clock is BRG2 010 SCC1 receive clock is BRG3 011 SCC1 receive clock is BRG4 100 SCC1 receive clock is CLK11 101 SCC1 receive clock is CLK12 110 SCC1 receive clock is CLK3 111 SCC1 receive clock is CLK4
5–7	TS1CS	Transmit SCC1 clock source (NMSI mode). Ignored if SCC1 is connected to the TSA (SC1 = 1) 000 SCC1 transmit clock is BRG1 001 SCC1 transmit clock is BRG2 010 SCC1 transmit clock is BRG3 011 SCC1 transmit clock is BRG4 100 SCC1 transmit clock is CLK11 101 SCC1 transmit clock is CLK12 110 SCC1 transmit clock is CLK3 111 SCC1 transmit clock is CLK4
8–15	—	Reserved, should be cleared
16	GR3	Grant support of SCC3 0 SCC3 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC3 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
17	SC3	SCC3 connection 0 SCC3 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC3 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.
18–20	RS3CS	Receive SCC3 clock source (NMSI mode). Ignored if SCC3 is connected to the TSA (SC3 = 1). 000 SCC3 receive clock is BRG1 001 SCC3 receive clock is BRG2 010 SCC3 receive clock is BRG3 011 SCC3 receive clock is BRG4 100 SCC3 receive clock is CLK5 101 SCC3 receive clock is CLK6 110 SCC3 receive clock is CLK7 111 SCC3 receive clock is CLK8

Table 24-5. CMXSCR Field Descriptions (continued)

Bits	Name	Description
21–23	TS3CS	Transmit SCC3 clock source (NMSI mode). Used as the USB clock when SCC3 is disabled or connected to the TSA (SC3 = 1) 000 SCC3 transmit/USB clock is BRG1 001 SCC3 transmit/USB clock is BRG2 010 SCC3 transmit/USB clock is BRG3 011 SCC3 transmit/USB clock is BRG4 100 SCC3 transmit/USB clock is CLK5 101 SCC3 transmit/USB clock is CLK6 110 SCC3 transmit/USB clock is CLK7 111 SCC3 transmit/USB clock is CLK8
24	GR4	Grant support of SCC4 0 SCC4 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC4 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
25	SC4	SCC4 connection 0 SCC4 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC4 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.
26–28	RS4CS	Receive SCC4 clock source (NMSI mode). Ignored if SCC4 is connected to the TSA (SC4 = 1) 000 SCC4 receive clock is BRG1 001 SCC4 receive clock is BRG2 010 SCC4 receive clock is BRG3 011 SCC4 receive clock is BRG4 100 SCC4 receive clock is CLK5 101 SCC4 receive clock is CLK6 110 SCC4 receive clock is CLK7 111 SCC4 receive clock is CLK8
29–31	TS4CS	Transmit SCC4 clock source (NMSI mode). Ignored if SCC4 is connected to the TSA (SC4 = 1) 000 SCC4 transmit clock is BRG1 001 SCC4 transmit clock is BRG2 010 SCC4 transmit clock is BRG3 011 SCC4 transmit clock is BRG4 100 SCC4 transmit clock is CLK5 101 SCC4 transmit clock is CLK6 110 SCC4 transmit clock is CLK7 111 SCC4 transmit clock is CLK8

24.4.5 CMX SMC Clock Route Register (CMXSMR)

The CMX SMC clock route register (CMXSMR), shown in Figure 24-9, defines the connection of the SMCs to the TSA and to the clock sources from the bank of clocks.

	0	1	2 3	4	5	6 7
Field	SMC1	—	SMC1CS	SMC2	—	SMC2CS
Reset	0000_0000					
R/W	R/W					
Offset	0x9_1B0C					

Figure 24-9. CMX SMC Clock Route Register (CMXSMR)

Table 24-6 describes CMXSMR fields.

Table 24-6. CMXSMR Field Descriptions

Bits	Name	Description
0	SMC1	SMC1 connection 0 SMC1 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SMCn pins is made in the parallel I/O control register. 1 SMC1 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
1	—	Reserved, should be cleared
2–3	SMC1CS	SMC1 clock source (NMSI mode). SMC1 can take its clocks from one of the two BRGs or one of two pins from the bank of clocks. However, the SMC1 transmit and receive clocks must be the same when it is connected to the NMSI. 00 SMC1 transmit and receive clocks are BRG1 01 SMC1 transmit and receive clocks are BRG7 10 SMC1 transmit and receive clocks are CLK7 11 SMC1 transmit and receive clocks are CLK9
4	SMC2	SMC2 connection 0 SMC2 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SMCn pins is made in the parallel I/O control register. 1 SMC2 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
5	—	Reserved, should be cleared
6–7	SMC2CS	SMC2 clock source (NMSI mode). SMC2 can take its clocks from one of the eight BRGs or one of eight pins from the bank of clocks. However, the SMC2 transmit and receive clocks must be the same when it is connected to the NMSI. 00 SMC2 transmit and receive clocks are BRG2 01 SMC2 transmit and receive clocks are BRG8 10 SMC2 transmit and receive clocks are CLK4 11 SMC2 transmit and receive clocks are CLK15

Chapter 25
Baud-Rate Generators (BRGs)

The CPM contains eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs, SCCs, and SMCs. The clocks produced by the BRGs are sent to the bank-of-clocks selection logic, where they can be routed to the controllers. In addition, the output of a BRG can be routed to a pin to be used externally. The following is a list of the main features of the BRGs in the CPM:

- Eight independent and identical BRGs
- On-the-fly changes allowed
- Each BRG can be routed to one or more FCCs, SCCs, or SMCs.
- A 16× divider option allows slow baud rates at high system frequencies.
- Each BRG contains an autobaud support option.
- Each BRG output can be routed to a pin (BRGO*n*).

Figure 25-1 shows a BRG.

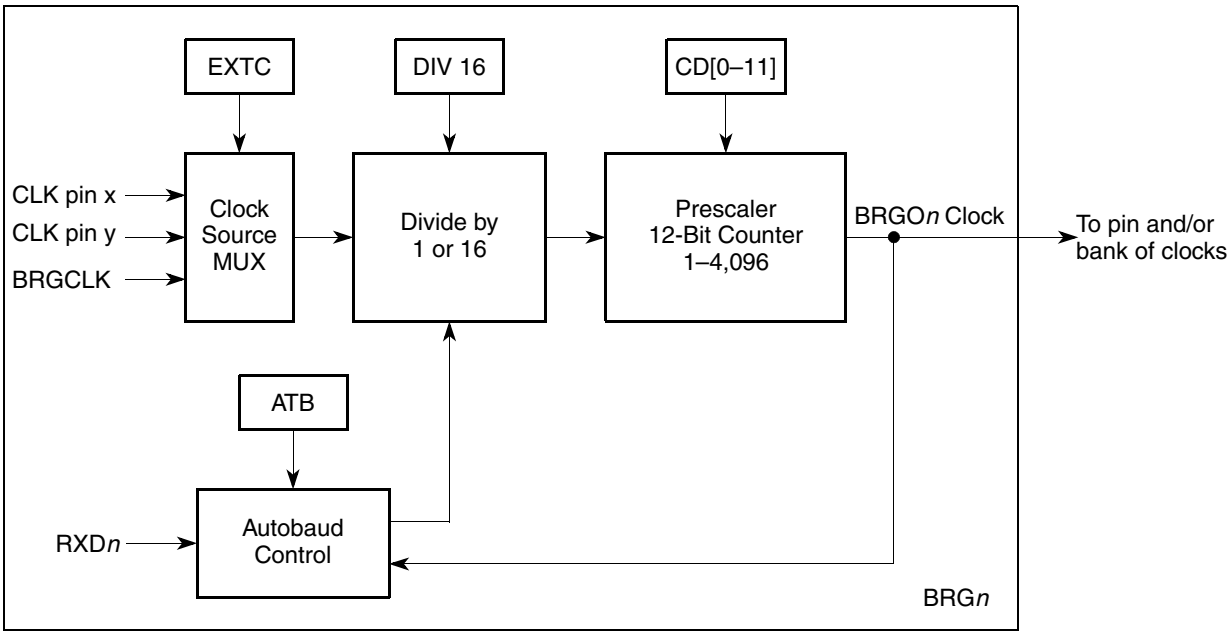


Figure 25-1. Baud-Rate Generator (BRG) Block Diagram

Each BRG clock source can be BRGCLK or a choice of two external clocks (selected in BRGC*n*[EXTC]). The BRGCLK is an internal signal generated in the MPC8555E clock synthesizer specifically for the BRGs, the SPI, and the I^2C internal BRG. Alternatively, external clock pins can be configured as clock sources. The external source option allows flexible baud-rate frequency generation, independent of the system frequency. Additionally, the external source option allows a single external frequency to be the

source for multiple BRGs. The external source signals are not synchronized internally before being used by the BRG.

The BRG provides a divide-by-16 option (BRGC*n*[DIV16]) and a 12-bit prescaler (BRGC*n*[CD]) to divide the source clock frequency. The combined source-clock divide factor can be changed on-the-fly; however, two changes should not occur within two source clock periods.

The prescaler output is sent internally to the bank of clocks and can also be output externally on BRGO*n* through the parallel I/O ports. If the BRG divides the clock by an even value, the transitions of BRGO*n* always occur on the falling edge of the source clock. If the divide factor is odd, the transitions alternate between the falling and rising edges of the source clock. Additionally, the output of the BRG can be sent to the autobaud control block.

25.1 System Clock Control Register (SCCR)

The system clock control register (SCCR), shown in Figure 25-2, is memory-mapped into the MPC8555E internal space. The SCCR defines the BRGCLK frequency to be supplied to the BRGs.

	0 — 15
Field	—
Reset	—
R/W	R/W
Offset	0x9_0C80

	16 — 29	30 31
Field	—	DFBRG
Reset	—	01
R/W	R/W	
Offset	0x9_0C82	

Figure 25-2. System Clock Control Register (SCCR)

Table 25-1 describes SCCR fields.

Table 25-1. SCCR Field Descriptions

Bits	Name	Defaults		Description
		POR	Hard Reset	
0–29	—			Reserved
30–31	DFBRG	01	Unaffected	Division factor of BRG_CLK relative to VCO_OUT (which is twice the CPM clock). Defines the BRG_CLK frequency. Changing the value does not result in a loss of lock condition. 00 Divide by 4 01 Divide by 16 (normal operation) 10 Divide by 64 11 Divide by 256

25.2 BRG Configuration Registers 1–8 (BRGC*n*)

The BRG configuration registers (BRGC*n*) are shown in Figure 25-3. A reset disables the BRG and drives the BRGO output clock high. The BRGC can be written at any time with no need to disable the SCCs or external devices that are connected to BRGO. Configuration changes occur at the end of the next BRG clock cycle (no spikes occur on the BRGO output clock). BRGC can be changed on-the-fly; however, two changes should not occur within a time equal to two source clock periods.

NOTE

Since the MPC8555E does not contain SCC2, ATB cannot function on BRG2.

	0 – 13	14	15
Field	—	RST	EN
Reset	0000_0000_0000_0000		
R/W	R/W		
Offset	0x9_19F0 (BRGC1), 0x9_19F4 (BRGC2), 0x9_19F8 (BRGC3), 0x9_19FC (BRGC4), 0x9_15F0 (BRGC5), 0x9_15F4 (BRGC6), 0x9_15F8 (BRGC7), 0x9_15FC (BRGC8)		

	16 – 17	18	19 – 30	31
Field	EXTC	ATB	CD	DIV16
Reset	0000_0000_0000_0000			
R/W	R/W			
Offset	0x9_19F2 (BRGC1), 0x9_19F6 (BRGC2), 0x9_19FA (BRGC3), 0x9_19FE (BRGC4), 0x9_15F2 (BRGC5), 0x9_15F6 (BRGC6), 0x9_15FA (BRGC7), 0x9_15FE (BRGC8)			

Figure 25-3. Baud-Rate Generator Configuration Registers (BRGC*n*)

Table 25-2 describes the BRGC*n* fields.

Table 25-2. BRGC*n* Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	RST	Reset BRG. Performs a software reset of the BRG identical to that of an external reset. A reset disables the BRG and drives BRGO high. This is externally visible only if BRGO is connected to the corresponding parallel I/O pin. 0 Enable the BRG. 1 Reset the BRG (software reset).
15	EN	Enable BRG count. Used to dynamically stop the BRG from counting—useful for low-power modes. 0 Stop all clocks to the BRG. 1 Enable clocks to the BRG.

Table 25-2. BRGC*n* Field Descriptions (continued)

Bits	Name	Description
16–17	EXTC	External clock source. Selects the BRG input clock. See Table 25-3. 00 The BRG input clock comes from the BRGCLK (internal clock generated from the CPM clock); see Section 25.1, "System Clock Control Register (SCCR)." 01 If BRG1, 2, 5, 6: The BRG input clock comes from the CLK3 pin. If BRG3, 4, 7, 8: The BRG input clock comes from the CLK9 pin. 10 If BRG1, 2, 5, 6: The BRG input clock comes from the CLK5 pin. If BRG3, 4, 7, 8: The BRG input clock comes from the CLK15 pin. 11 Reserved
18	ATB	Autobaud. Selects autobaud operation of the BRG on the corresponding RXD. ATB must remain zero until the SCC receives the 3 Rx clocks. Then the user must set ATB to obtain the correct baud rate. After the baud rate is obtained and locked, it is indicated by setting AB in the UART event register. 0 Normal operation of the BRG 1 When RXD goes low, the BRG determines the length of the start bit and synchronizes the BRG to the actual baud rate.
19–30	CD	Clock divider. CD presets an internal 12-bit counter that is decremented at the DIV16 output rate. When the counter reaches zero, it is reloaded with CD. CD = 0xFFF produces the minimum clock rate for BGRO (divide by 4,096); CD = 0x000 produces the maximum rate (divide by 1). When dividing by an odd number, the counter ensures a 50% duty cycle by asserting the terminal count once on clock low and next on clock high. The terminal count signals counter expiration and toggles the clock. See Section 25.4, "UART Baud Rate Examples."
31	DIV16	Divide-by-16. Selects a divide-by-1 or divide-by-16 prescaler before reaching the clock divider. See Section 25.4, "UART Baud Rate Examples." 0 Divide by 1 1 Divide by 16

Table 25-3 shows the possible external clock sources for the BRGs.

Table 25-3. BRG External Clock Source Options

BRG	CLK																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
BRG1			V		V															
BRG2			V		V															
BRG3									V						V					
BRG4									V						V					
BRG5			V		V															
BRG6			V		V															
BRG7									V						V					
BRG8									V						V					

25.3 Autobaud Operation on a UART

During the autobaud process, a UART deduces the baud rate of its received character stream by examining the received pattern and its timing. A built-in autobaud control function automatically measures the length of a start bit and modifies the baud rate accordingly.

If the autobaud bit BRGC*n*[ATB] is set, the autobaud control function starts searching for a low level on the corresponding RXD*n* input, which it assumes marks the beginning of a start bit, and begins counting the start bit length. During this time, the BRG output clock toggles for 16 BRG clock cycles at the BRG source clock rate and stops with BRGO*n* in the low state.

When RXD*n* goes high again, the autobaud control block rewrites BRGC*n*[CD, DIV16] to the divide ratio found, which at high baud rates may not be exactly the final rate desired (for example, 56,600 may result rather than 57,600). An interrupt can be enabled in the UART SCC event register to report that the autobaud controller rewrote BRGC*n*. The interrupt handler can then adjust BRGC*n*[CD, DIV16] (see Table 25-4) for accuracy before the first character is fully received, ensuring that the UART recognizes all characters.

After a full character is received, the software can verify that the character matches a predefined value (such as ‘a’ or ‘A’). Software should then check for other characters (such as ‘t’ or ‘T’) and program the preferred parity mode in the UART’s protocol-specific mode register (PSMR).

Note that the SCC associated with this BRG must be programmed to UART mode and select the 16× option for TDCR and RDCR in the general SCC mode register low. Input frequencies such as 1.8432, 3.68, 7.36, and 14.72 MHz should be used. The SCC performing the autobaud function must be connected to that SCC’s BRG; that is, SCC3 must be clocked by BRG3, and so on.

Also, to detect an autobaud lock and generate an interrupt, the SCC must receive 3 full Rx clocks from the BRG before the autobaud process begins. To do this, first clear BRGC*n*[ATB] and enable the BRG Rx clock to the highest frequency. Then, immediately before the autobaud process starts (after device initialization), set BRGC*n*[ATB].

25.4 UART Baud Rate Examples

For synchronous communication using the internal BRG, the BRGO must not exceed the BRG input clock divided by 2. Therefore, with a BRG input clock of 66 MHz (generated using an external clock source, refer to BRGC*n*[EXTC]), the maximum BRGO rate is 33 MHz. Program the UART to 16× oversampling when using the SCC as a UART. Rates of 8× and 32× are also available. Assuming 16× oversampling is chosen in the UART, the maximum data rate is 66 MHz ÷ 16 = 4.125 Mbps. Keeping the above in mind, use the following formula to calculate the bit rate based on a particular BRG configuration for a UART:

$$\text{Async Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \times (\text{Clock Divider} + 1) \times (\text{Sampling Rate})}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \times (\text{BRGCx[CD]} + 1) \times (\text{GSMRx\_L[xDCR]})}$$

Table 25-4 lists typical bit rates of asynchronous communication. Note that here the internal clock rate is assumed to be 16× the baud rate; that is, GSMR*x*_L[TDCR] = GSMR*x*_L[RDCR] = 0b10.

Table 25-4. Typical Baud Rates for Asynchronous Communication

Baud Rate	Using a 66-MHz BRG Input Clock		
	BRGC*n*[DIV16]	BRGC*n*[CD]	Actual Frequency (Hz)
75	1	3436	75.01
150	1	1718	149.98
300	1	858	300.13
600	1	429	599.56
1200	0	3436	1200.2
2400	0	1718	2399.7
4800	0	858	4802.1
9600	0	429	9593.0
19,200	0	214	19,186
38,400	0	106	38,551
57,600	0	71	57,292
115,200	0	35	114,583
460,000	0	8	458,333

For synchronous communication, the internal clock is identical to the baud-rate output. To get the preferred rate, select the system clock according to the following:

$$\text{Sync Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \times (\text{Clock Divider} + 1)}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \times (\text{BRGCx[CD]} + 1)}$$

For example, to get a rate of 64 Kbps, the system clock can be 24.96 MHz, BRGC*n*[DIV16] = 0, and BRGC*n*[CD] = 389.

Chapter 26
CPM Timers

The CPM includes four identical 16-bit general-purpose timers or two 32-bit timers. Each general-purpose timer consists of a timer mode register (TMR), a timer capture register (TCR), a timer counter (TCN), a timer reference register (TRR), a timer event register (TER), and a timer global configuration register (TGCR). The TMRs contain the prescalar values programmed by the user.

Figure 26-1 shows the timer block diagram.

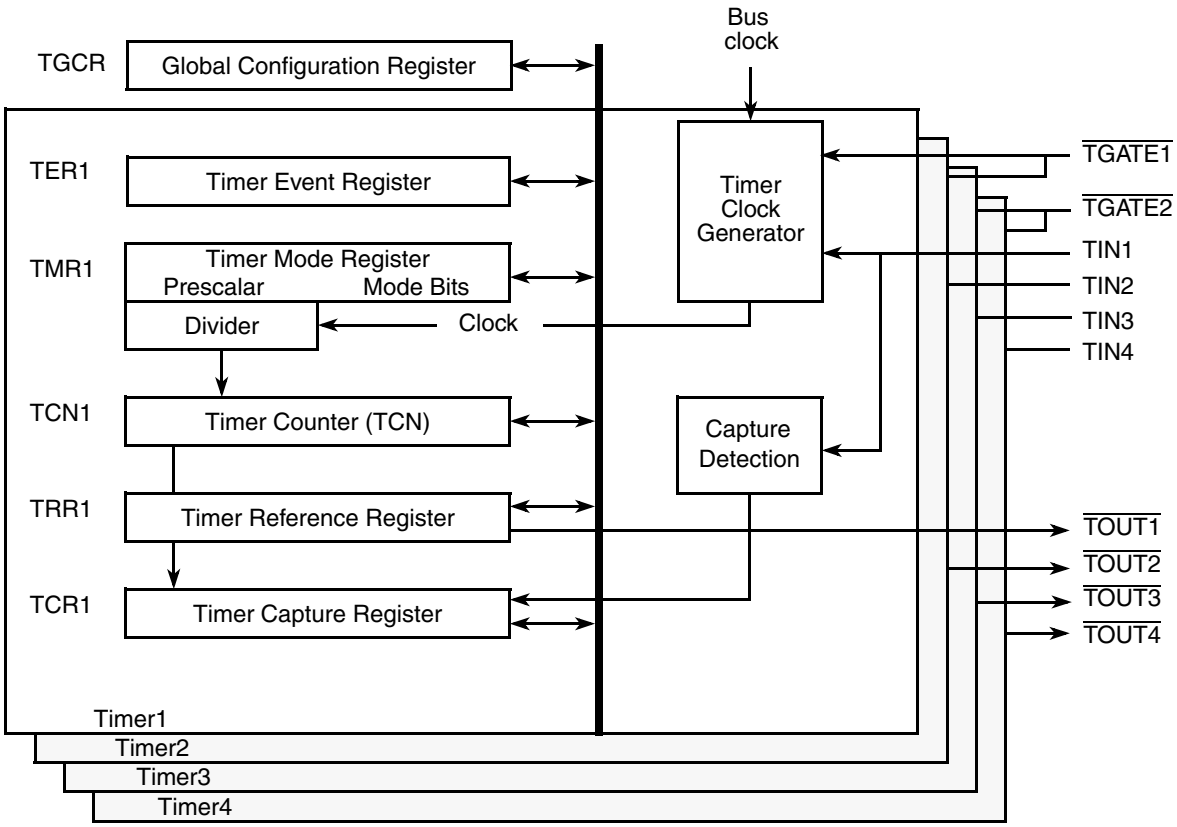


Figure 26-1. Timer Block Diagram

Pin assignments for TIN*x*, $\overline{\text{TGATE}x}$, and $\overline{\text{TOUT}x}$ are described in Section 45.5, "Port Tables."

26.1 Features

The key features of the CPM timers include the following:

- The maximum input clock is the internal CPM clock which is the core complex bus (CCB) clock divided by 3.

- Maximum period of 2.6 seconds (at 100 MHz)
- 10-ns resolution (at 100 MHz)
- Programmable sources for the clock input
- Input capture capability
- Output compare with programmable mode for the output pin
- Two timers cascade internally or externally to form a 32-bit timer
- Free run and restart modes
- Functional compatibility with timers on the MC68360, MPC860, and MPC8260

26.2 General-Purpose Timer Units

The clock input to the prescaler can be selected from the following three sources:

- Internal CPM clock (CCB/3)
- Internal CPM clock divided by 16 (CCB/48)
- Corresponding TIN*x*, programmed in the parallel port registers

The internal CPM clock is generated in the CPM clock synthesizer and defaults to the CCB clock frequency divided by 3. 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*x* to be the clock source. TIN*x* is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding TMR[ICLK] bits. The prescalar is programmed to divide the clock input by values from 1 to 256 and the output of the prescalar is used as an input to the 16-bit counter. The best resolution of the timer is one clock cycle (10 ns at 100 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (2.6 seconds at 100 MHz).

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 TMR selects each mode. Upon reaching the reference value, the corresponding TER bit is set and an interrupt is issued if TMR[ORI] = 1. The timers can output a signal on the timer outputs ($\overline{\text{TOUT1}}$–$\overline{\text{TOUT4}}$) when the reference value is reached (selected by the corresponding TMR[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-bit timer.

In addition, each timer has a 16-bit TCR used to latch the value of the counter when a defined transition of TIN1, TIN2, TIN3, or TIN4 is sensed by the corresponding input capture edge detector. The type of transition triggering the capture is selected by the corresponding TMR[CE] bits. Upon a capture or reference event, the corresponding TER bit is set and a maskable interrupt request is issued to the interrupt controller. The timers may be gated/restarted by an external gate signal. There are two gate signals—$\overline{\text{TGATE1}}$ controls timer 1 and/or 2 and $\overline{\text{TGATE2}}$ controls timer 3 and/or 4. Normal gate mode enables the count on a falling edge of $\overline{\text{TGATE}x}$ and disables the count on the rising edge of $\overline{\text{TGATE}x}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}x}$.

The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of $\overline{\text{TGATE}x}$. This mode has applications in pulse interval measurement and bus monitoring as follows:

- Pulse measurement—The restart gate mode can measure a low $\overline{\text{TGATE}x}$. The rising edge of $\overline{\text{TGATE}x}$ completes the measurement and if $\overline{\text{TGATE}x}$ is connected externally to TIN*x*, 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 abnormally stuck low. The bus signal should be connected to $\overline{\text{TGATE}x}$. 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 TMR; the gate operating mode is selected in the TGCR.

NOTE

$\overline{\text{TGATE}x}$ is internally synchronized to the internal CPM clock. After the falling edge of $\overline{\text{TGATE}x}$ is recognized, the counter begins counting after one internal CPM clock cycle when working with the internal clock.

26.2.1 Cascaded Mode

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 can be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user can select two 16-bit timers or one 32-bit timer. TGCR is used to put the timers into cascaded mode, as shown in Figure 26-2.

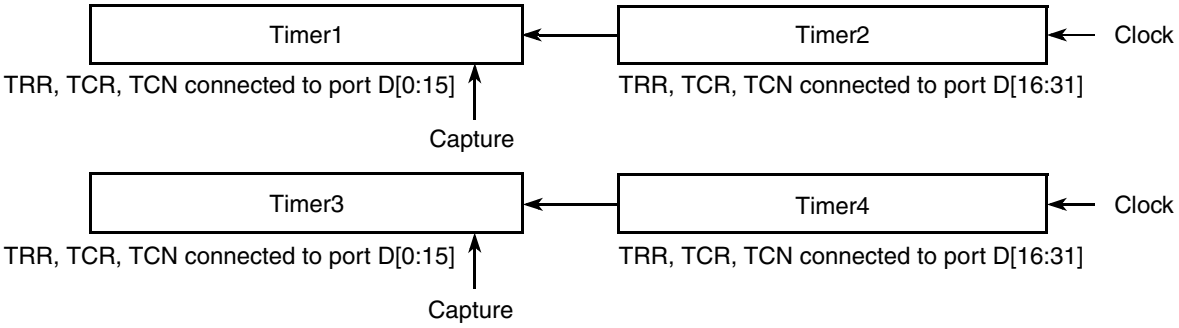


Figure 26-2. Timer Cascaded Mode Block Diagram

If TGCR[CAS] = 1, the two timers function as a 32-bit timer with a 32-bit TRR, TCR, and TCN. In this case, TMR1 and/or TMR3 are ignored, and the modes are defined using TMR2 and/or TMR4. The capture is controlled from TIN2 or TIN4 and the interrupts are generated from TER2 or TER4. In cascaded mode, the combined TRR, TCR, and TCN must be referenced with 32-bit bus cycles.

26.2.2 Timer Global Configuration Registers (TGCR1, TGCR2)

The timer global configuration registers (TGCR1 and TGCR2), shown in Figure 26-3 and Figure 26-4, contain configuration parameters used by the timers. These registers allow simultaneous starting and stopping of a pair of timers (1 and 2 or 3 and 4) if one bus cycle is used.

	0	1	2	3	4	5	6	7
Field	CAS2	—	STP2	RST2	GM1	—	STP1	RST1
Reset	0000_0000							
R/W	R/W							
Offset	0x9_0D80							

Figure 26-3. Timer Global Configuration Register 1 (TGCR1)

Table 26-1 describes TGCR1 fields.

Table 26-1. TGCR1 Field Descriptions

Bits	Name	Description
0	**CAS2**	Cascade timers 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer.
1	—	Reserved, should be cleared.
2	**STP 2**	Stop timer 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user 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 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP bit is cleared.
4	**GM1**	Gate mode for $\overline{\text{TGATE1}}$. This bit is valid only if the gate function is enabled in TMR1 or TMR2. 0 Restart gate mode. $\overline{\text{TGATE1}}$ is used to enable/disable count. A falling $\overline{\text{TGATE1}}$ enables and restarts the count and a rising edge of $\overline{\text{TGATE1}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE1}}$ does not restart the count value in TCN.
5	—	Reserved, should be cleared.
6	**STP1**	Stop timer 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user 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	**RST1**	Reset timer 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if STP = 0.

The TGCR2 register is shown in Figure 26-4.

	0	1	2	3	4	5	6	7
Field	CAS4	—	STP4	RST4	GM2	—	STP3	RST3
Reset	0000_0000							
R/W	R/W							
Offset	0x9_0D84							

Figure 26-4. Timer Global Configuration Register 2 (TGCR2)

Table 26-2 describes TGCR2 fields.

Table 26-2. TGCR2 Field Descriptions

Bit	Name	Description
0	**CAS4**	Cascade timers 0 Normal operation 1 Timers 3 and 4 cascades to form a 32-bit timer.
1	—	Reserved, should be cleared.
2	**STP 4**	Stop timer 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user 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 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP bit is cleared.
4	**GM2**	Gate mode for $\overline{\text{TGATE2}}$. This bit is valid only if the gate function is enabled in TMR3 or TMR4. 0 Restart gate mode. $\overline{\text{TGATE2}}$ is used to enable/disable the count. The falling edge of $\overline{\text{TGATE2}}$ enables and restarts the count and the rising edge of $\overline{\text{TGATE2}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE2}}$ does not restart the count value in TCN.
5	—	Reserved, should be cleared.
6	**STP3**	Stop timer 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, however it is possible to read the values while the clock is stopped. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	**RST3**	Reset timer 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if STP = 0.

26.2.3 Timer Mode Registers (TMR1–TMR4)

The four timer mode registers (TMR1–TMR4) are shown in Figure 26-5.

Erratic behavior may occur if TGCR1 and TGCR2 are not initialized before the TMRs. Only TGCR[RST] can be modified at any time.

	0 7	8 9	10	11	12	13 14	15
Field	PS	CE	OM	ORI	FRR	ICLK	GE
Reset	0000_0000_0000_0000						
R/W	R/W						
Offset	0x9_0D90 (TMR1); 0x9_0D92 (TMR2); 0x9_0DA0 (TMR3); 0x9_0DA2 (TMR4)						

Figure 26-5. Timer Mode Registers (TMR1–TMR4)

Table 26-3 describes TMR1–TMR4 register fields.

Table 26-3. TMR1–TMR4 Field Descriptions

Bits	Name	Description
0–7	**PS**	Prescalar value. The prescalar is programmed to divide the clock input by values from 1 to 256. The value 00000000 divides the clock by 1 and 1111_1111 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*x* edge only and enable interrupt on capture event. 10 Capture on falling TIN*x* edge only and enable interrupt on capture event. 11 Capture on any TIN*x* edge and enable interrupt on capture event.
10	**OM**	Output mode 0 Active-low pulse on $\overline{\text{TOUT}x}$ for one timer input clock cycle as defined by the ICLK bits. Thus, $\overline{\text{TOUT}x}$ may be low for one bus clock period, one bus clock/16 period, or one TIN*x* clock cycle period. $\overline{\text{TOUT}x}$ changes occur on the rising edge of the system clock. 1 Toggle $\overline{\text{TOUT}x}$. $\overline{\text{TOUT}x}$ changes occur on 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 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.
13–14	**ICLK**	Input clock source for the timer 00 Internally cascaded input. For TMR1, the timer 1 input is the output of timer 2. For TMR3, the timer 3 input is the output of timer 4. For TMR2 and TMR4, this selection means no input clock is provided to the timer. 01 Internal bus clock 10 Internal bus clock divided by 16 11 Corresponding TIN*x*: TIN1, TIN2, TIN3, or TIN4 (falling edge).
15	**GE**	Gate enable 0 $\overline{\text{TGATE}x}$ is ignored. 1 $\overline{\text{TGATE}x}$ is used to control the timer.

26.2.4 Timer Reference Registers (TRR1–TRR4)

Each timer reference register (TRR1–TRR4), shown in Figure 26-6, contains the timeout's reference value. The reference value is not reached until TCN*x* increments to equal the timeout reference value.

	0	15
Field	Timeout reference value	
Reset	0xFFFF	
R/W	R/W	
Offset	0x9_0D94 (TRR1), 0x9_0D96 (TRR2), 0x9_0DA4 (TRR3), 0x9_0DA6 (TRR4)	

Figure 26-6. Timer Reference Registers (TRR1–TRR4)

26.2.5 Timer Capture Registers (TCR1–TCR4)

Each timer capture register (TCR1–TCR4), shown in Figure 26-7, is used to latch the value of the counter according to TMR*x*[CE].

	0	15
Field	Latched counter value	
Reset	0x0000	
R/W	R/W	
Offset	0x9_0D98 (TCR1), 0x9_0D9A (TCR2), 0x9_0DA8 (TCR3), 0x9_0DAA (TCR4)	

Figure 26-7. Timer Capture Registers (TCR1–TCR4)

26.2.6 Timer Counters (TCN1–TCN4)

Each timer counter register (TCN1–TCN4), shown in Figure 26-8, is an up-counter. A read cycle to TCN*x* yields the current value of the timer but does not affect the counting operation. A write cycle to TCN*x* sets the register to the written value, thus causing its corresponding prescalar, TMR*x*[PS], to be reset.

	0	15
Field	Up counter	
Reset	0x0000	
R/W	R/W	
Offset	0x9_0D9C (TCN1), 0x9_0D9E (TCN2), 0x9_0DAC (TCN3), 0x9_0DAE (TCN4)	

Figure 26-8. Timer Counter Registers (TCN1–TCN4)

Note that the counter registers may not be updated correctly if a write is made while the timer is not running. Use TRR*x* to define the preferred count value.

26.2.7 Timer Event Registers (TER1–TER4)

Each timer event register (TER*x*), shown in Figure 26-9, reports events recognized by the timers. When an output reference event is recognized, the timer sets TER*x*[REF] regardless of the corresponding TMR*x*[ORI]. The capture event is set only if it is enabled by TMR*x*[CE]. TER1–TER4 can be read at any time.

Writing ones clears event bits; writing zeros has no effect. Both event bits must be cleared before the timer negates the interrupt.

	0	13	14	15
Field	—		REF	CAP
Reset	0x0000			
Offset	0x9_0DB0 (TER1); 0x9_0DB2 (TER2); 0x9_0DB4 (TER3); 0x9_0DB6 (TER4)			

Figure 26-9. Timer Event Registers (TER1–TER4)

Table 26-4 describes TER fields.

Table 26-4. TER Field Descriptions

Bits	Name	Description
0–13	–	Reserved, should be cleared.
14	REF	Output reference event. The counter has reached the TRR value. TMR[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Capture event. The counter value has been latched into the TCR. TMR[CE] is used to enable generation of this event.

Chapter 27
SDMA Channels

The MPC8555E has two physical serial DMA (SDMA) channels. The CP implements two dedicated virtual SDMA channels for each FCC, SCC, SMC, SPI, USB, and I^2C—one for each transmitter and receiver.

Table 27-1 shows data flow paths. Data from the peripheral controllers can be routed to external DDR SDRAM using the DDR bus (path 1) or the local bus (path 2).

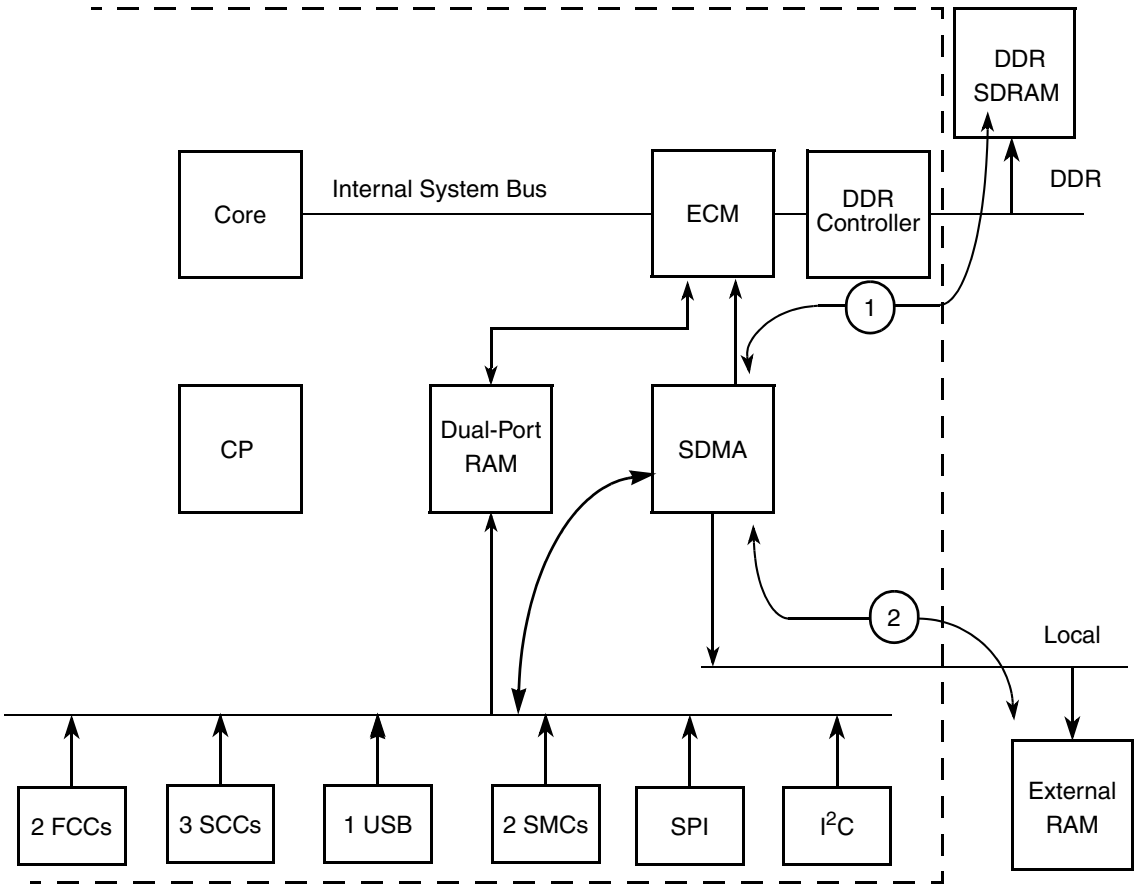


Figure 27-1. SDMA Data Paths

On a path 1 access, the SDMA channel must acquire the external DDR system bus. On a path 2 access, the local bus is acquired and the access is not seen on the external DDR system bus. Thus, the local bus transfer occurs at the same time as other operations on the external DDR system bus.

The SDMA channel always operates in big-endian format when accesses data.

If a system or local bus error occurs on a CP-related access by the SDMA, the CP generates an interrupt which will set the SDMA SYS or SDMA LCL bit in the SIPNR_L register. Section 22.5.1.3, “CPM Interrupt Pending Registers (SIPNR_H, SIPNR_L).” The interrupt service routine then reads the appropriate DMA transfer error address register (SMAER for the system bus or LMAER for the local bus) to determine the address the bus error occurred on. The channel that caused the bus error is determined by reading the channel number from SMEVR or LMEVR.

27.1 SDMA Registers

27.1.1 SDMA Address Error Registers (SMAER, LMAER)

These registers indicate the address of the error that occurred at a read or write transaction initiated by the system or local channel SDMA. The SMAER holds the system address of the system SDMA initiated error transaction, and the LMAER holds the local address of the local SDMA initiated error transaction. SMAER and LMAER are cleared by hard reset. They are not affected by soft reset. These registers are updated at the first occurrence of the error detection; the next capture is enabled only after the event register is cleared.

	0 — 31
Field	SMAER/LMAER
R/W	R
Reset	0000_0000_0000_0000_0000_0000_0000_0000
Offset	0x9_0050 (SMAER); 0x9_0060 (LMAER)

Figure 27-2. SDMA Address Error Registers (SMAER and LMAER)

27.1.2 SDMA Event Registers (SMEVR, LMEVR)

These registers indicate if the error information that is found in SMAER/LMAER is valid. They are updated in the first SDMA data error that is detected. Bit 0 is cleared by writing 1 to it. Writing 0 has no effect. MSNUM contains the serial number of the channel that created the data error. The bit fields of these two registers are updated only when a read error occurs. SMEVR and LMEVR are cleared by hard reset. Soft reset has no effect.

	0	1	2 — 7	8 — 31
Field	DATAERR	—	MSNUM	—
R/W	R/W		R	R/W
Reset	0000_0000_0000_0000_0000_0000_0000_0000			
Offset	0x9_0058 (SMEVR); 0x9_0068 (LMEVR)			

Figure 27-3. SDMA Event Registers (SMEVR and LMEVR)

Table 27-1. SMEVR and LMEVR Field Descriptions

Bits	Name	Description
0	DATAERR	Indicates Data error, Error details are captured in SMAER/LMAER registers, and in MSNUM field. Bit is cleared by writing ‘1’ to it. Writing ’0’ has no effect. 0 No error occurred 1 Error occurred while reading / writing data
1	—	Reserved
2–7	MSNUM	Represent the serial number of the channel that created the data error
8–31	—	Reserved

27.1.3 SDMA Control Registers (SMCTR, LMCTR)

The SDMA control registers (SMCTR and LMCTR), shown in Figure 27-4, contain control parameters and can be updated through SREG.

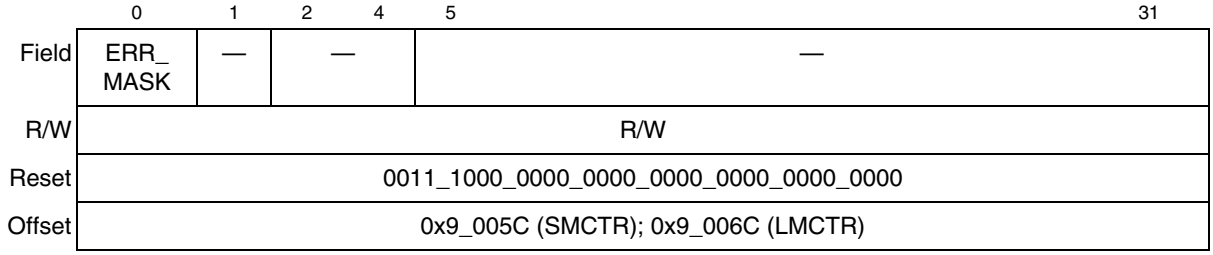


Figure 27-4. SDMA Control Registers (SMCTR and LMCTR)

Table 27-2. SMCTR/LMCTR Field Descriptions

Bits	Name	Description
0	ERR_MASK	Data error interrupt mask 0 Disable error interrupt 1 Enable error interrupt
1	—	Reserved
2–4	—	Reserved **Note:** User must not change default value of bits 2–4 (‘111’) when writing to SMCTR/LMCTR
5–31	—	Reserved

Chapter 28
Serial Communications Controllers (SCCs)

The MPC8555E has three serial communications controllers (SCCs), which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks. An SCC has many physical interface options such as interfacing to TDM buses, ISDN buses, and standard modem interfaces.

The SCCs are independent from the physical interface, but SCC logic formats and manipulates data from the physical interface. Furthermore, the choice of protocol is independent from the choice of interface. An SCC is described in terms of the protocol it runs. When an SCC is programmed to a certain protocol or mode, it implements functionality that corresponds to parts of the protocol's link layer (layer 2 of the OSI reference model). Many SCC functions are common to protocols of the following controllers:

- UART, described in Chapter 29, "SCC UART Mode"
- HDLC and HDLC bus, described in Chapter 30, "SCC HDLC Mode"
- AppleTalk/LocalTalk, described in Chapter 33, "SCC AppleTalk Mode"
- BISYNC, described in Chapter 31, "SCC BISYNC Mode"
- Transparent, described in Chapter 32, "SCC Transparent Mode"
- Although the selected protocol usually applies both to the SCC transmitter and receiver, one half of an SCC can run transparent operations while the other runs a standard protocol.

Each Rx and Tx internal clock can be programmed with either an external or internal source. Internal clocks originate from one of eight baud rate generators (BRGs) or an external clock pin; see Section 24.3, "NMSI Configuration," for each SCC's available clock sources. These clocks can be as fast as a 1:4 ratio of the system clock. (For example, an SCC internal clock can run at 12.5 MHz in a 50-MHz system.) However, an SCC's ability to support a sustained bit stream depends on the protocol as well as other factors.

NOTE

This clock ratio is based on the hardware architecture and does not ensure that an application will run at that speed. It is the responsibility of the system designer to check AC specifications of the I/O pins and determine the maximum frequency.

Associated with each SCC is a digital phase-locked loop (DPLL) for external clock recovery, which supports NRZ, NRZI, FM0, FM1, Manchester, and Differential Manchester. If the clock recovery function is not required (that is, synchronous communication), then the DPLL can be disabled, in which case only NRZ and NRZI are supported.

An SCC can be connected to its own set of pins on the MPC8555E. This configuration is called the non-multiplexed serial interface (NMSI) and is described in Chapter 23, "Serial Interface with Time-Slot

Assigner." Using NMSI, an SCC can support standard modem interface signals, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. If required, software and additional parallel I/O lines can be used to support additional handshake signals. Figure 28-1 shows the SCC block diagram.

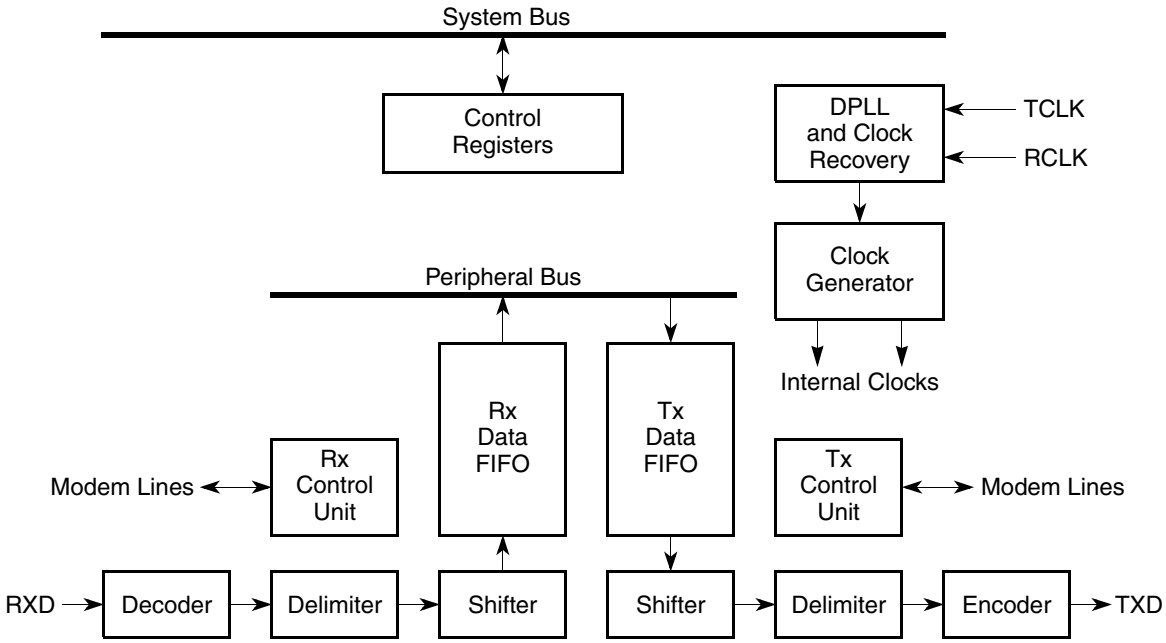


Figure 28-1. SCC Block Diagram

28.1 Features

The following is a list of the main SCC features. (Performance figures assume a 25-MHz system clock.)

- Implements HDLC/SDLC, HDLC bus, synchronous start/stop, asynchronous start/stop (UART), AppleTalk/LocalTalk, and totally transparent protocols
- Additional protocols may be available through the use of RAM-based microcodes. Please contact your Freescale sales office.
- DPLL circuitry for clock recovery with NRZ, NRZI, FM0, FM1, Manchester, and Differential Manchester (also known as Differential Bi-phase-L)
- Clocks can be derived from a baud rate generator, an external pin, or DPLL
- Supports automatic control of the $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem signals
- Multi-buffer data structure for receive and send (the number of buffer descriptors (BDs) is limited only by the size of the internal dual-port RAM—8 bytes per BD)
- Deep FIFOs (SCC transmit and receive FIFOs are 32 bytes each.)
- Transmit-on-demand feature decreases time to frame transmission (transmit latency)
- Low FIFO latency option for send and receive in character-oriented and totally transparent protocols
- Frame preamble options
- Full-duplex operation

- Fully transparent option for one half of an SCC (Rx/Tx) while another protocol executes on the other half (Tx/Rx)
- Echo and local loopback modes for testing

28.2 General SCC Mode Registers (GSMR1–GSMR4)

Each SCC contains a general SCC mode register (GSMR) that defines options common to most of the protocols. GSMR_L contains the low-order 32 bits; GSMR_H, shown in Figure 28-2, contains the high-order 32 bits. Some GSMR operations are described in later sections.

	0	14	15
Field	—		
Reset	0000_0000_0000_0000		
R/W	R/W		
Offset	0x9_1A04 (GSMR1); 0x9_1A44 (GSMR3); 0x9_1A64 (GSMR4)		

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TCRC		REVD	TRX	TTX	CDP	CTSP	CDS	CTSS	TFL	RFW	TXSY	SYNL		RTSM	RSYN
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1A06 (GSMR1); 0x9_1A46 (GSMR3); 0x9_1A66 (GSMR4)															

Figure 28-2. GSMR_H—General SCC Mode Register (High Order)

Table 28-1 describes GSMR_H fields.

Table 28-1. GSMR_H Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–17	TCRC	Transparent CRC (valid for totally transparent channel only). Selects the frame checking provided on transparent channels of the SCC (either the receiver, transmitter, or both, as defined by TTX and TRX). Although this configuration selects a frame check type, the decision to send the frame check is made in the TxBD. Thus, frame checks are not needed in transparent mode and frame check errors generated on the receiver can be ignored. 00 16-bit CCITT CRC (HDLC). (X16 + X12 + X5 + 1) 01 CRC16 (BISYNC). (X16 + X15 + X2 + 1) 10 32-bit CCITT CRC (HDLC) (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 + 1) 11 Reserved
18	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation 1 Reverses the bit order for totally transparent channels on this SCC (either the receiver, transmitter, or both) and sends the msb of each byte first. Section 31.11, “BISYNC Mode Register (PSMR),” describes reversing bit order in a BISYNC protocol.

Table 28-1. GSMR_H Field Descriptions (continued)

Bits	Name	Description
19–20	TRX, TTX	Transparent receiver/transmitter. The receiver, transmitter, or both can use totally transparent operation, regardless of GSMR_L[MODE]. For example, to configure the transmitter as a UART and the receiver for totally transparent operations, set MODE = 0b0100 (UART), TTX = 0, and TRX = 1. 0 Normal operation 1 The channel uses totally transparent mode, regardless of the protocol chosen in GSMR_L[MODE]. For full-duplex totally transparent operation, set both TTX and TRX.
21, 22	CDP, CTSP	$\overline{CD}$/$\overline{CTS}$ pulse. If this SCC is used in the TSA and is programmed in transparent mode, set CTSP and refer to Section 32.4.2, "Synchronization and the TSA," for options on programming CDP. 0 Normal operation (envelope mode). $\overline{CD}$/$\overline{CTS}$ should envelope the frame. Negating $\overline{CD}$/$\overline{CTS}$ during reception causes a $\overline{CD}$/$\overline{CTS}$ lost error. 1 Pulse mode. Synchronization occurs when $\overline{CD}$/$\overline{CTS}$ is asserted; further $\overline{CD}$/$\overline{CTS}$ transitions do not affect reception.
23, 24	CDS, CTSS	$\overline{CD}$/$\overline{CTS}$ sampling. Determine synchronization characteristics of $\overline{CD}$ and $\overline{CTS}$. If the SCC is in transparent mode and is used in the TSA, CDS and CTSS must be set. Also, CDS and CTSS must be set for loopback testing in transparent mode. 0 $\overline{CD}$/$\overline{CTS}$ is assumed to be asynchronous with data. It is internally synchronized by the SCC, then data is received ($\overline{CD}$) or sent ($\overline{CTS}$) after several clock delays. 1 $\overline{CD}$/$\overline{CTS}$ is assumed to be synchronous with data, which speeds up operation. $\overline{CD}$ or $\overline{CTS}$ must transition while the Rx/Tx clock is low, at which time, the transfer begins. Useful for connecting MPC8555E in transparent mode since the $\overline{RTS}$ of one MPC8555E can connect directly to the $\overline{CD}$/$\overline{CTS}$ of another.
25	TFL	Transmit FIFO length 0 Normal operation. The transmit FIFO is 32 bytes. 1 The Tx FIFO is 1 byte. This option is used with character-oriented protocols, such as UART, to ensure a minimum FIFO latency at the expense of performance.
26	RFW	Rx FIFO width 0 Receive FIFO is 32 bits wide for maximum performance; the Rx FIFO is 32 bytes. Data is not normally written to receive buffers until at least 32 bits are received. This configuration is required for HDLC-type protocols and is recommended for high-performance transparent protocols. 1 Low-latency operation. The receive FIFO is 8 bits wide, reducing the Rx FIFO to a quarter its normal size. This allows data to be written to the buffer as soon as a character is received, instead of waiting to receive 32 bits. This configuration must be chosen for character-oriented protocols, such as UART. It can also be used for low-performance, low-latency, transparent operation. However, it must not be used with HDLC, HDLC Bus, or AppleTalk because it causes erratic behavior.
27	TXSY	Transmitter synchronized to the receiver. Intended for X.21 applications where the transmitted data must begin an exact multiple of 8-bit periods after the received data arrives. 0 No synchronization between receiver and transmitter (default). 1 The transmit bit stream is synchronized to the receiver. Additionally, if RSYN = 1, transmission in totally transparent mode does not occur until the receiver synchronizes with the bit stream and $\overline{CTS}$ is asserted to the SCC. Assuming $\overline{CTS}$ is asserted, transmission begins 8 clocks after the receiver starts receiving data.
28–29	SYNL	Sync length (BISYNC and transparent mode only). See the data synchronization register (DSR) definition in Section 31.9, "Sending and Receiving the Synchronization Sequence," (BISYNC) and Section 32.4.1.1, "In-Line Synchronization Pattern," (transparent). 00 An external sync ($\overline{CD}$) is used instead of the sync pattern in the DSR. 01 4-bit sync. The receiver synchronizes on a 4-bit sync pattern stored in the DSR. This sync and additional syncs can be stripped by programming the SCC's parameter RAM for character recognition. 10 8-bit sync. Should be chosen along with the BISYNC protocol to implement mono-sync. The receiver synchronizes on an 8-bit sync pattern in the DSR. 11 16-bit sync. Also called BISYNC. The receiver synchronizes on a 16-bit sync pattern stored in the DSR.

Table 28-1. GSMR_H Field Descriptions (continued)

Bits	Name	Description
30	RTSM	$\overline{\text{RTS}}$ mode. Determines whether flags or idles are to be sent. Can be changed on-the-fly. 0 Send idles between frames as defined by the protocol and the TEND bit. $\overline{\text{RTS}}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{\text{RTS}}$ is always asserted whenever the SCC is enabled.
31	RSYN	Receive synchronization timing (totally transparent mode only) 0 Normal operation. 1 If CDS = 1, $\overline{\text{CD}}$ should be asserted on the second bit of the Rx frame rather than on the first.

Figure 28-3 shows GSMR_L.

	0	1	2	3	4	5	6	7	8	10	11	12	13	14	15
Field	—	EDGE		TCI	TSNC		RINV	TINV	TPL		TPP		TEND	TDCR	
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_1A00 (SCC1); 0x9_1A40 (SCC3); 0x9_1A60 (SCC4)														

	16	17	18	20	21	23	24	25	26	27	28	31
Field	RDCR		RENC		TENC		DIAG		ENR	ENT	MODE	
Reset	0000_0000_0000_0000											
R/W	R/W											
Offset	0x9_1A02 (SCC1); 0x9_1A42 (SCC3); 0x9_1A62 (SCC4)											

Figure 28-3. GSMR_L—General SCC Mode Register (Low Order)

Table 28-2 describes GSMR_L fields.

Table 28-2. GSMR_L Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1–2	EDGE	Clock edge. Determines the clock edge the DPLL uses to adjust the receive sample point due to jitter in the received signal. Ignored in UART protocol or if the 1x clock mode is selected in RDCR. 00 Both the positive and negative edges are used for changing the sample point (default). 01 Positive edge. Only the positive edge of the received signal is used to change the sample point. 10 Negative edge. Only the negative edge of the received signal is used to change the sample point. 11 No adjustment is made to the sample point.

Table 28-2. GSMR_L Field Descriptions (continued)

Bits	Name	Description
3	TCI	Transmit clock invert 0 Normal operation 1 Before it is used, the internal Tx clock (TCLK) is inverted by the SCC so it can clock data out one-half clock earlier (on the rising rather than the falling edge). In this case, the SCC offers a minimum and maximum rising clock edge-to-data specification. Data output by the SCC after the rising edge of an external Tx clock can be latched by the external receiver one clock cycle later on the next rising edge of the same Tx clock. In HDLC and Transparent mode, when TCI = 0, data is sent on the falling edge; when TCI = 1, on the rising edge. Recommended for HDLC and transparent operation when clock rates exceed 8 MHz to improve data setup time for the external transceiver.
4–5	TSNC	Transmit sense. Determines the amount of time the internal carrier sense signal stays active after the last transition on RXD, indicating that the line is free. For instance, AppleTalk can use TSNC to avoid a spurious CS-changed (SCCE[DCC]) interrupt that would otherwise occur during the frame sync sequence before the opening flags. If RDCR is configured to 1× clock mode, the delay is the greater of the two numbers listed. If RDCR is configured to 8×, 16×, or 32× mode, the delay is the smaller number. 00 Infinite. Carrier sense is always active (default) 01 14- or 6.5-bit times as determined by RDCR 10 4- or 1.5-bit times as determined by RDCR (normally for AppleTalk) 11 3- or 1-bit times as determined by RDCR
6	RINV	DPLL Rx input invert data. Must be zero in HDLC bus mode or asynchronous UART mode 0 Do not invert. 1 Invert data before sending it to the DPLL for reception. Used to produce FM1 from FM0 and NRZI space from NRZI mark or to invert the data stream in regular NRZ mode.
7	TINV	DPLL Tx input invert data. Must be zero in HDLC bus mode 0 Do not invert. 1 Invert data before sending it to the DPLL for transmission. Used to produce FM1 from FM0 and NRZI space from NRZI mark and to invert the data stream in regular NRZ mode. In T1 applications, setting TINV and TEND creates a continuously inverted HDLC data stream.
8–10	TPL	Tx preamble length. Determines the length of the preamble configured by the TPP bits 000 No preamble (default) 001 8 bits (1 byte) 010 16 bits (2 bytes) 011 32 bits (4 bytes) 100 48 bits (6 bytes) 101 64 bits (8 bytes) 110 128 bits (16 bytes) 111 Reserved
11–12	TPP	Tx preamble pattern. Determines what, if any, bit pattern should precede each Tx frame. The preamble pattern is sent before the first flag/sync of the frame. TPP is ignored in UART mode. The preamble length is programmed in TPL; the preamble pattern is typically sent to a receiving station that uses a DPLL for clock recovery. The receiving DPLL uses the regular preamble pattern to help it lock onto the received signal in a short, predictable time period. 00 All zeros 01 Repetitive 10s 10 Repetitive 01s 11 All ones. Select this setting for LocalTalk operation.

Table 28-2. GSMR_L Field Descriptions (continued)

Bits	Name	Description
13	TEND	Transmitter frame ending. Intended for NRZI transmitter encoding of the DPLL. TEND determines whether TXD should idle in a high state or in an encoded ones state (high or low). It can, however, be used with other encodings besides NRZI. 0 Default operation. TXD is encoded only when data is sent, including the preamble and opening and closing flags/syncs. When no data is available to send, the signal is driven high. 1 TXD is always encoded, even when idles are sent.
14–15	TDCR	Transmitter/receiver DPLL clock rate. If the DPLL is not used, choose 1× mode except in asynchronous UART mode where 8×, 16×, or 32× must be chosen. TDCR should match RDCR in most applications to allow the transmitter and receiver to use the same clock source. If an application uses the DPLL, the selection of TDCR/RDCR depends on the encoding/decoding. If communication is synchronous, select 1×. FM0/FM1, Manchester, and Differential Manchester require 8×, 16×, or 32×. If NRZ- or NRZI-encoded communication is asynchronous (that is, clock recovery required), select 8×, 16×, or 32×. The 8× option allows highest speed, whereas the 32× option provides the greatest resolution. 00 1× clock mode. Only NRZ or NRZI encodings/decodings are allowed. 01 8× clock mode 10 16× clock mode. Normally chosen for UART and AppleTalk 11 32× clock mode
16–17	RDCR	
18–20	RENC	Receiver decoding/transmitter encoding method. Select NRZ if DPLL is not used. RENC should equal TENC in most applications. 000 NRZ (default setting if DPLL is not used). Required for UART (synchronous or asynchronous) 001 NRZI Mark (set RINV/TINV also for NRZI space) 010 FM0 (set RINV/TINV also for FM1) 011 Reserved 100 Manchester 101 Reserved 110 Differential Manchester (Differential Bi-phase-L) 111 Reserved
21–23	TENC	
24–25	DIAG	Diagnostic mode 00 Normal operation, $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are under automatic control. Data is received through RXD and transmitted through TXD. The SCC uses modem signals to enable or disable transmission and reception. These timings are shown in Section 28.4.5, “Controlling SCC Timing with RTS, CTS, and CD.” 01 Local loopback mode. Transmitter output is connected internally to the receiver input, while the receiver and the transmitter operate normally. The value on RXD is ignored. If enabled, data appears on TXD, or the parallel I/O registers can be programmed to make TXD high. $\overline{\text{RTS}}$ can also be programmed to be disabled in the appropriate parallel I/O register. The transmitter and receiver must share the same clock source, but separate CLK*x* pins can be used if connected to the same external clock source. If external loopback is preferred, program DIAG for normal operation and externally connect TXD and RXD. Then, physically connect the control signals ($\overline{\text{RTS}}$ connected to $\overline{\text{CD}}$, and $\overline{\text{CTS}}$ grounded) or set the parallel I/O registers so $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are permanently asserted to the SCC by configuring the associated $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins as general-purpose I/O. 10 Automatic echo mode. The transmitter automatically resends received data bit-by-bit using the Rx clock provided. The receiver operates normally and receives data if $\overline{\text{CD}}$ is asserted. $\overline{\text{CTS}}$ is ignored. 11 Loopback and echo mode. Loopback and echo operation occur simultaneously. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are ignored. See the loopback bit description above for clocking requirements. For TDM operation, the diagnostic mode is selected by SI*x*MR[SDM*x*]; see Section 23.6.2, “SI Mode Registers (SI2MR).”

Table 28-2. GSMR_L Field Descriptions (continued)

Bits	Name	Description
26	ENR	Enable receive. Enables the receiver hardware state machine for this SCC. 0 The receiver is disabled and data in the Rx FIFO is lost. If ENR is cleared during reception, the receiver aborts the current character. 1 The receiver is enabled. ENR can be set or cleared, regardless of whether serial clocks are present. Section 28.4.7, "Reconfiguring the SCCs," describes how to disable/enable an SCC. Note that other tools, including the ENTER HUNT MODE and CLOSE RXBD commands and the E bit of the RxBD, data provide the capability to control the receiver.
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this SCC. 0 The transmitter is disabled. If ENT is cleared during transmission, the current character is aborted and TXD returns to the idle state. Data already in the Tx shift register is not sent. 1 The transmitter is enabled. ENT can be set or cleared, regardless of whether serial clocks are present. Section 28.4.7, "Reconfiguring the SCCs," describes how to disable/enable an SCC. Note that other tools, such as the STOP TRANSMIT, GRACEFUL STOP TRANSMIT, and RESTART TRANSMIT commands, the freeze option and $\overline{\text{CTS}}$ flow control option in UART mode, and the R bit of the TxBD, also provide the capability to control the transmitter.
28–31	MODE	Channel protocol mode. See also GSMR_H[TTX, TRX]. 0000 HDLC 0001 Reserved 0010 AppleTalk/LocalTalk 0011 SS7—reserved for RAM microcode 0100 UART 0101 Profibus—reserved for RAM microcode 0110 Reserved 0111 Reserved 1000 BISYNC 1001 Reserved 1010 QMC 1011 Reserved 1100 Reserved 11xx Reserved

28.2.1 Protocol-Specific Mode Register (PSMR)

The protocol implemented by an SCC is selected by its GSMR_L[MODE]. Each SCC has an additional protocol-specific mode register (PSMR) that configures it specifically for the chosen protocol. The PSMR fields are described in the specific chapters that describe each protocol. PSMRs are cleared at reset. PSMRs reside at the following addresses: 0x9_1A08 (PSMR1), 0x9_1A28 (PSMR2), 0x9_1A48 (PSMR3), and 0x9_1A68 (PSMR4).

28.2.2 Data Synchronization Register (DSR)

Each SCC has a data synchronization register (DSR) that specifies the pattern used for frame synchronization. The programmed value for DSR depends on the following protocols:

- UART—DSR is used to configure fractional stop bit transmission.
- BISYNC and transparent—DSR should be programmed with the sync pattern.
- HDLC—At reset, DSR defaults to 0x7E7E (two HDLC flags), so it does not need to be written.

Figure 28-4 shows the sync fields.

	0 – 7	8 – 15
Field	SYN2	SYN1
Reset	0111_1110	0111_1110
R/W	R/W	
Offset	0x9_1A0E (DSR1); 0x9_1A2E (DSR2); 0x9_1A4E (DSR3); 0x9_1A6E (DSR4)	

Figure 28-4. Data Synchronization Register (DSR)

28.2.3 Transmit-on-Demand Register (TODR)

In normal operation, if no frame is being sent by an SCC, the CP periodically polls the R bit of the next TxBD to see if a new frame/buffer is requested. Depending on the SCC configuration, this polling occurs every 8–32 serial Tx clocks. The transmit-on-demand option, selected in the transmit-on-demand register (TODR) shown in Figure 28-5, shortens the latency of the Tx buffer/frame and is useful in LAN-type protocols where maximum inter-frame gap times are limited by the protocol specification.

	0	1 – 15
Field	TOD	—
Reset	0000_0000_0000_0000	
R/W	R/W	
Offset	0x9_1A0C (TODR1); 0x9_1A2C (TODR2); 0x9_1A4C (TODR3); 0x9_1A6C (TODR4)	

Figure 28-5. Transmit-on-Demand Register (TODR)

The CP can be configured to begin processing a new frame/buffer without waiting the normal polling time by setting TODR[TOD] after TxBD[R] is set. Because this feature favors the specified TxBD, it may affect servicing of other SCC FIFOs. Therefore, transmitting on demand should only be used when a high-priority TxBD has been prepared and enough time has passed since the last g transmission. Table 28-3 describes TODR fields.

Table 28-3. TODR Field Descriptions

Bits	Name	Description
0	TOD	Transmit on demand 0 Normal operation 1 The CP gives high priority to the current TxBD and begins sending the frame without waiting the normal polling time to check the TxBD's R bit. TOD is cleared automatically after one serial clock, but transmitting on demand continues until an unprepared (R = 0) BD is reached. TOD does not need to be set again if new TxBDs are added to the BD table as long as older TxBDs are still being processed. New TxBDs are processed in order. The first bit of the frame is typically clocked out 5-6 bit times after TOD is set.
1–15	—	Reserved, should be cleared.

28.3 SCC Buffer Descriptors (BDs)

Data associated with each SCC channel is stored in buffers and each buffer is referenced by a buffer descriptor (BD) that can reside anywhere in dual-port RAM. The total number of 8-byte BDs is limited only by the size of the dual-port RAM (128 BDs/1 Kbyte). These BDs are shared among all serial controllers—SCCs, SMCs, SPI, and I^2C. The user defines how the BDs are allocated among the controllers.

Each 64-bit BD has the following structure:

- The half word at offset + 0x0 contains status and control bits that control and report on the data transfer. These bits vary from protocol to protocol. The CPM updates the status bits after the buffer is sent or received.
- The half word at offset + 0x2 (data length) holds the number of bytes sent or received.
  - For an RxBD, this is the number of bytes the controller writes into the buffer. The CPM writes the length after received data is placed into the associated buffer and the buffer closed. In frame-based protocols (but not including SCC transparent operation), this field contains the total frame length, including CRC bytes. Also, if a received frame's length, including CRC, is an exact multiple of MRBLR, the last BD holds no actual data but does contain the total frame length.
  - For a TxBD, this is the number of bytes the controller should send from its buffer. Normally, this value should be greater than zero. The CPM never modifies this field.
- The word at offset + 0x4 (buffer pointer) points to the beginning of the buffer in memory (internal or external).
  - For an RxBD, the value must be a multiple of four. (word-aligned)
  - For a TxBD, this pointer can be even or odd.

Shown in Figure 28-6, the format of Tx and RxBDs is the same in each SCC mode. Only the status and control bits differ for each protocol.

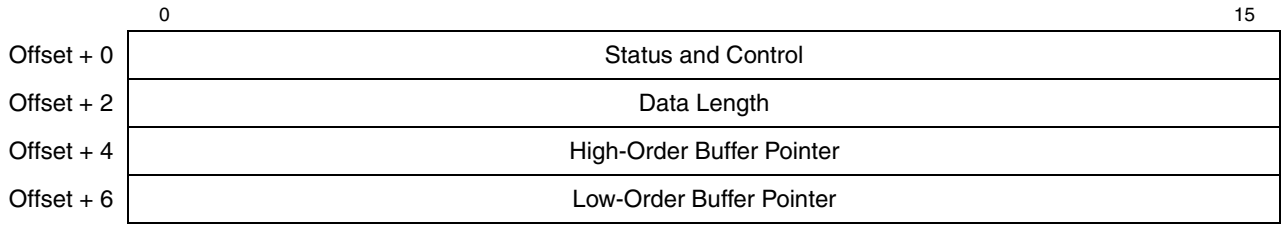


Figure 28-6. SCC Buffer Descriptors (BDs)

For frame-oriented protocols, a message can reside in as many buffers as necessary. Each buffer has a maximum length of 65,535 bytes. The CPM does not assume that all buffers of a single frame are currently linked to the BD table. The CPM does assume, however, that the unlinked buffers are provided by the core in time to be sent or received; otherwise, an error condition is reported—an underrun error when sending and a busy error when receiving. Figure 28-7 shows the SCC BD table and buffer structure.

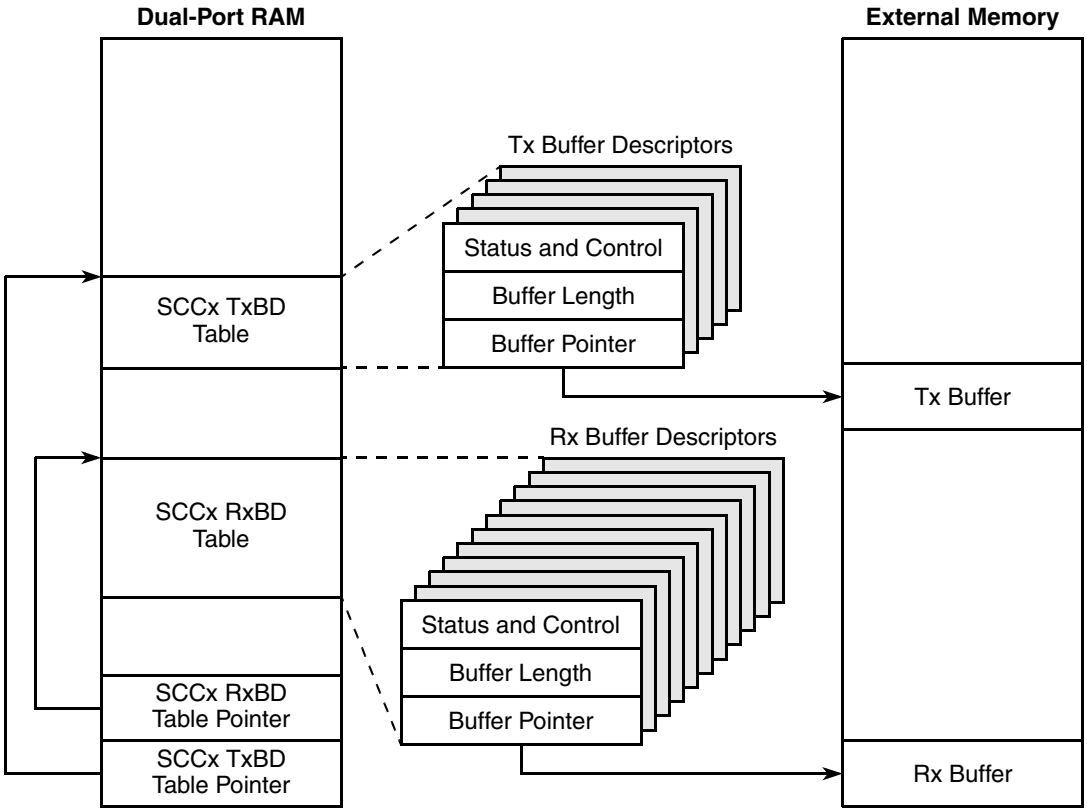


Figure 28-7. SCC BD and Buffer Memory Structure

In all protocols, BDs can point to buffers in the internal dual-port RAM. However, because dual-port RAM is used for descriptors, buffers are usually put in external RAM, especially if they are large.

The CPM processes TxBDs straightforwardly; when the transmit side of an SCC is enabled, the CPM starts with the first BD in that SCC TxBD table. Once the CPM detects that the R bit is set in the TxBD, it starts processing the buffer. The CPM detects that the BD is ready when it polls the R bit or when the user writes to the TODR. After data from the BD is put in the Tx FIFO, if necessary the CPM waits for the next descriptor's R bit to be set before proceeding. Thus, the CPM does no look-ahead descriptor processing and does not skip BDs that are not ready. When the CPM sees a BD's W bit (wrap) set, it returns to the start of the BD table after this last BD of the table is processed. The CPM clears R (not ready) after using a TxBD, which keeps it from being retransmitted before it is confirmed by the core. However, some protocols support a continuous mode (CM), for which R is not cleared (always ready).

The CPM uses RxBDs similarly. When data arrives, the CPM performs required processing on the data and moves resultant data to the buffer pointed to by the first BD; it continues until the buffer is full or an event, such as an error or end-of-frame detection, occurs. The buffer is then closed; subsequent data uses the next BD. If E = 0, the current buffer is not empty and it reports a busy error. The CPM does not move from the current BD until E is set by the core (the buffer is empty). After using a descriptor, the CPM clears E (not empty) and does not reuse a BD until it has been processed by the core. However, in continuous mode (CM), E remains set. When the CPM discovers a descriptor's W bit set (indicating it is the last BD in the circular BD table), it returns to the beginning of the table when it is time to move to the next buffer.

28.4 SCC Parameter RAM

Each SCC parameter RAM area begins at the same offset from each SCC base area. Section 28.4.1, "SCC Base Addresses," describes the SCC's base addresses. The protocol-specific portions of the SCC parameter RAM are discussed in the specific protocol descriptions and the part that is common to all SCC protocols is shown in Table 28-4.

Some parameter RAM values must be initialized before the SCC can be enabled. Other values are initialized or written by the CPM. Once initialized, most parameter RAM values do not need to be accessed because most activity centers around the descriptors rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled—after a STOP TRANSMIT command and before a RESTART TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command.
- Rx parameter RAM can be written only when the receiver is disabled. Note the CLOSE RXBD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.
- See Section 28.4.7, "Reconfiguring the SCCs."

Table 28-4 shows the parameter RAM map for all SCC protocols. Boldfaced entries must be initialized by the user.

Table 28-4. SCC Parameter RAM Map for All Protocols

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address—offset from the beginning of dual-port RAM. The BD tables can be placed in any unused portion of the dual-port RAM. The CPM starts BD processing at the top of the table. (The user defines the end of the BD table by setting the W bit in the last BD to be processed.) Initialize these entries before enabling the corresponding channel. Erratic operations occur if BD tables of active SCCs overlap. Values in RBASE and TBASE should be multiples of eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx function code. See Section 28.4.2, "Function Code Registers (RFCR, TFCR)."
0x05	**TFCR**	Byte	Tx function code. See Section 28.4.2, "Function Code Registers (RFCR, TFCR)."
0x06	**MRBLR**	Hword	Maximum receive buffer length. Defines the maximum number of bytes the MPC8555E writes to a receive buffer before it goes to the next buffer. The MPC8555E can write fewer bytes than MRBLR if a condition such as an error or end-of-frame occurs. It never writes more bytes than the MRBLR value. Therefore, user-supplied buffers should be no smaller than MRBLR. MRBLR should be greater than zero for all modes. It should be a multiple of 4 for HDLC mode, and in totally transparent mode unless the Rx FIFO is 8-bits wide (GSMR_H[RFW] = 1). Note that although MRBLR is not intended to be changed while the SCC is operating, it can be changed dynamically in a single-cycle, 16-bit move (not two 8-bit cycles). Changing MRBLR has no immediate effect. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the receiver is disabled. Transmit buffer length is programmed in TxBD[Data Length] and is not affected by MRBLR.
0x08	RSTATE	Word	Rx internal state[3]

Table 28-4. SCC Parameter RAM Map for All Protocols (continued)

Offset [1]	Name	Width	Description
0x0C	—	Word	Rx internal buffer pointer[2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	Current RxBD pointer. Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the BD table is reached, the CPM initializes RBPTR to the value in the RBASE. Although most applications do not need to write RBPTR, it can be modified when the receiver is disabled or when no Rx buffer is in use.
0x12	—	Hword	Rx internal byte count [2]. The Rx internal byte count is a down-count value initialized with MRBLR and decremented with each byte written by the supporting SDMA channel.
0x14	—	Word	Rx temp[3]
0x18	TSTATE	Word	Tx internal state[3]
0x1C		Word	Tx internal buffer pointer [2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	Current TxBD pointer. Points to the current BD being processed or to the next BD the transmitter uses when it is idling. After reset or when the end of the BD table is reached, the CPM initializes TBPTR to the value in the TBASE. Although most applications do not need to write TBPTR, it can be modified when the transmitter is disabled or when no Tx buffer is in use (after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission).
0x22	—	Hword	Tx internal byte count [2]. A down-count value initialized with TxBD[Data Length] and decremented with each byte read by the supporting SDMA channel.
0x24	—	Word	Tx temp[3]
0x28	RCRC	Word	Temp receive CRC [2]
0x2C	TCRC	Word	Temp transmit CRC [2]
0x30	—	—	Protocol-specific area. (The size of this area depends on the protocol chosen.)

1 From SCC base. See Section 28.4.1, “SCC Base Addresses.”
2 These parameters need not be accessed for normal operation but may be helpful for debugging.
3 For CP use only.

28.4.1 SCC Base Addresses

The CPM maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the FCCs, SCCs, SPI, and I^{2}C. SCC base addresses are described in Table 28-5.

The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM.

Table 28-5. Parameter RAM—SCC Base Addresses

Page	Address[1]	Peripheral	Size (Bytes)
1	0x8000	SCC1	256
2	0x8100	Reserved	256
3	0x8200	SCC3	256
4	0x8300	SCC4	256

[1] Offset from RAM_Base.

28.4.2 Function Code Registers (RFCR, TFCR)

There are six separate function code registers for the three SCC channels, three for Rx buffers (RFCR1, RFCR3, and RFCR4) and three for Tx buffers (TFCR1, TFCR3, and TFCR4). The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 28-8 shows the register format.

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**—**	**DTB**	—
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	SCC*x* base + 0x04 (RFCR*x*); SCC*x* base + 0x05 (TFCR*x*)							

Figure 28-8. Function Code Registers (RFCR and TFCR)

Table 28-6 describes RFCR*x*/TFCR*x* fields.

Table 28-6. RFCR*x*/TFCR*x* Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (HDLC and transparent) or at the beginning of the next BD. 0x Reserved 1x Big endian
5	**—**	Reserved, should be cleared
6	**DTB**	Data bus indicator 0 Use system bus for SDMA operation 1 Use local bus for SDMA operation
7	—	Reserved, should be cleared.

28.4.3 Handling SCC Interrupts

To allow interrupt handling for SCC-specific events, event, mask, and status registers are provided within each SCC's internal memory map area; see Table 28-7. Because interrupt events are protocol-dependent, event descriptions are found in the specific protocol chapters.

Table 28-7. SCC*x* Event, Mask, and Status Registers

Register Offset	Description
SCCE*x* 0x9_1A10 (SCCE1); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4)	SCC event register. This 16-bit register reports events recognized by any of the SCCs. When an event is recognized, the SCC sets its corresponding bit in SCCE, regardless of the corresponding mask bit. When the corresponding event occurs, an interrupt is signaled to the SIVEC register. Bits are cleared by writing ones (writing zeros has no effect). SCCE is cleared at reset and can be read at any time.
SCCM*x* 0x9_1A14 (SCCM1); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)	SCC mask register. The 16-bit, read/write register allows interrupts to be enabled or disabled using the CPM for specific events in each SCC channel. An interrupt is generated only if SCC interrupts in this channel are enabled in the CPM interrupt mask register (SIMR). If an SCCM bit is zero, the CPM does not proceed with interrupt handling when that event occurs. The SCCM and SCCE bit positions are identical.
SCCS*x* 0x9_1A17 (SCCS1); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)	SCC status register. This 8-bit, read-only register allows monitoring of the real-time status of RXD.

Follow these steps to handle an SCC interrupt:

1. When an interrupt occurs, read SCCE to determine the interrupt sources and clear those SCCE bits (in most cases).
2. Process the TxBDs to reuse them if SCCE[TX] or SCCE[TXE] = 1. If the transmit speed is fast or the interrupt delay is long, the SCC may have sent more than one Tx buffer. Thus, it is important to check more than one TxBD during interrupt handling. A common practice is to process all TxBDs in the handler until one is found with its R bit set.
3. Extract data from the RxBD if SCCE[RX], SCCE[RXB], or SCCE[RXF] is set. As with transmit buffers, if the receive speed is fast or the interrupt delay is long, the SCC may have received more than one buffer and the handler should check more than one RxBD. A common practice is to process all RxBDs in the interrupt handler until one is found with RxBD[E] set.
4. Execute the **rfi** instruction.

For additional information about interrupt handling refer to Chapter 22, "CPM Interrupt Controller."

28.4.4 Initializing the SCCs

The SCCs require that a number of registers and parameters be configured after a power-on reset. Regardless of the protocol used, follow these steps to initialize SCCs:

1. Write the parallel I/O ports to configure and connect the I/O pins to the SCCs.
2. Configure the parallel I/O registers to enable $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ if these signals are required.
3. If the time-slot assigner (TSA) is used, the serial interface (SI*x*) must be configured. If the SCC is used in NMSI mode, CMXSCR must still be initialized.

4. Write all GSMR bits except ENT or ENR.
5. Write the PSMR.
6. Write the DSR.
7. Initialize the required values for this SCC's parameter RAM.
8. Initialize the transmit/receive parameters through the CP command register (CPCR).
9. Clear out any current events in SCCE (optional).
10. Write ones to SCCM register to enable interrupts.
11. Set GSMR_L[ENT] and GSMR_L[ENR].

Descriptors can have their R or E bits set at any time. Notice that the CPCR does not need to be accessed after a hardware reset. An SCC should be disabled and re-enabled after any dynamic change to its parallel I/O ports or serial channel physical interface configuration. A full reset can also be implemented using CPCR[RST].

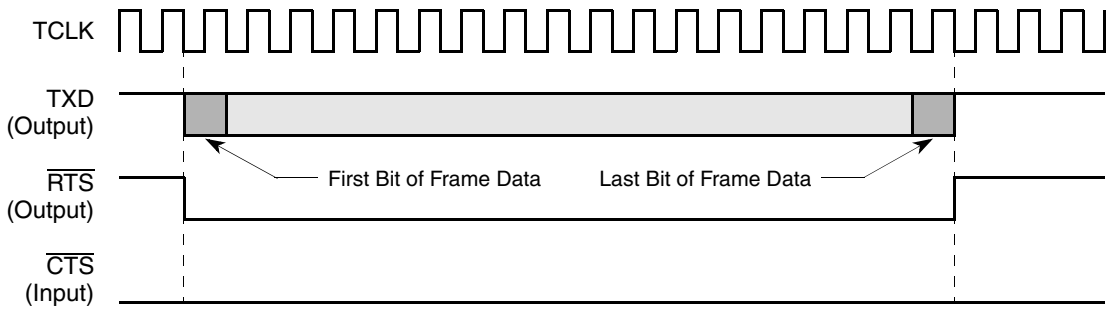
28.4.5 Controlling SCC Timing with $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$

When GSMR_L[DIAG] is programmed to normal operation, $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are controlled by the SCC. In the following subsections, it is assumed that GSMR_L[TCI] is zero, implying normal transmit clock operation.

28.4.5.1 Synchronous Protocols

$\overline{\text{RTS}}$ is asserted when the SCC data is loaded into the Tx FIFO and a falling Tx clock occurs. At this point, the SCC starts sending data once appropriate conditions occur on $\overline{\text{CTS}}$. In all cases, the first data bit is the start of the opening flag, sync pattern, or preamble.

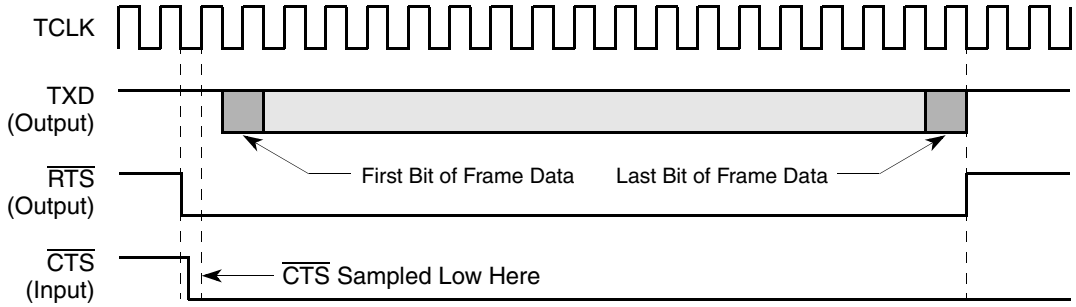
Figure 28-9 shows that the delay between $\overline{\text{RTS}}$ and data is 0 bit times, regardless of GSMR_H[CTSS]. This operation assumes that $\overline{\text{CTS}}$ is already asserted to the SCC or that $\overline{\text{CTS}}$ is reprogrammed to be a parallel I/O line, in which case $\overline{\text{CTS}}$ to the SCC is always asserted. $\overline{\text{RTS}}$ is negated one clock after the last bit in the frame.



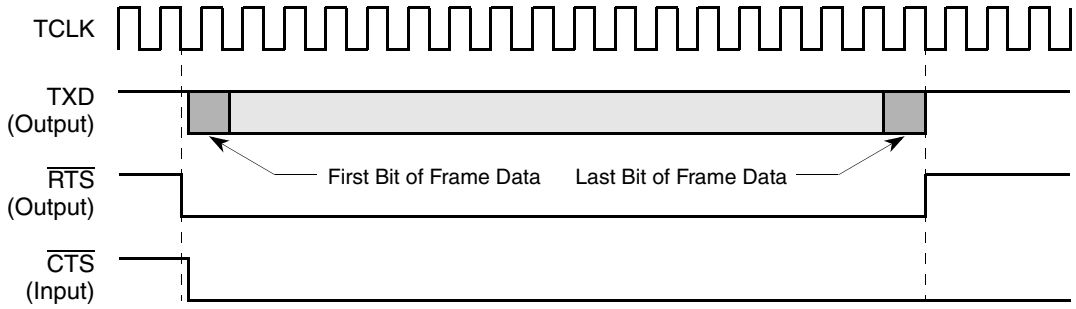
Note:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 28-9. Output Delay from $\overline{\text{RTS}}$ Asserted for Synchronous Protocols

When $\overline{\text{RTS}}$ is asserted, if $\overline{\text{CTS}}$ is not already asserted, delays to the first data bit depend on when $\overline{\text{CTS}}$ is asserted. Figure 28-10 shows that the delay between $\overline{\text{CTS}}$ and the data can be approximately 0.5 to 1 bit times or no delay, depending on GSMR_H[CTSS].



Note:
1. GSMR_H[CTSS] = 0. CTSP is a do not care.



Note:
1. GSMR_H[CTSS] = 1. CTSP is a do not care.

Figure 28-10. Output Delay from $\overline{\text{CTS}}$ Asserted for Synchronous Protocols

If $\overline{\text{CTS}}$ is programmed to envelope data, negating it during frame transmission causes a $\overline{\text{CTS}}$ lost error. Negating $\overline{\text{CTS}}$ forces $\overline{\text{RTS}}$ high and Tx data to become idle. If GSMR_H[CTSS] is zero, the SCC must sample $\overline{\text{CTS}}$ before a $\overline{\text{CTS}}$ lost is recognized; otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the $\overline{\text{CTS}}$ lost condition. See Figure 28-11.

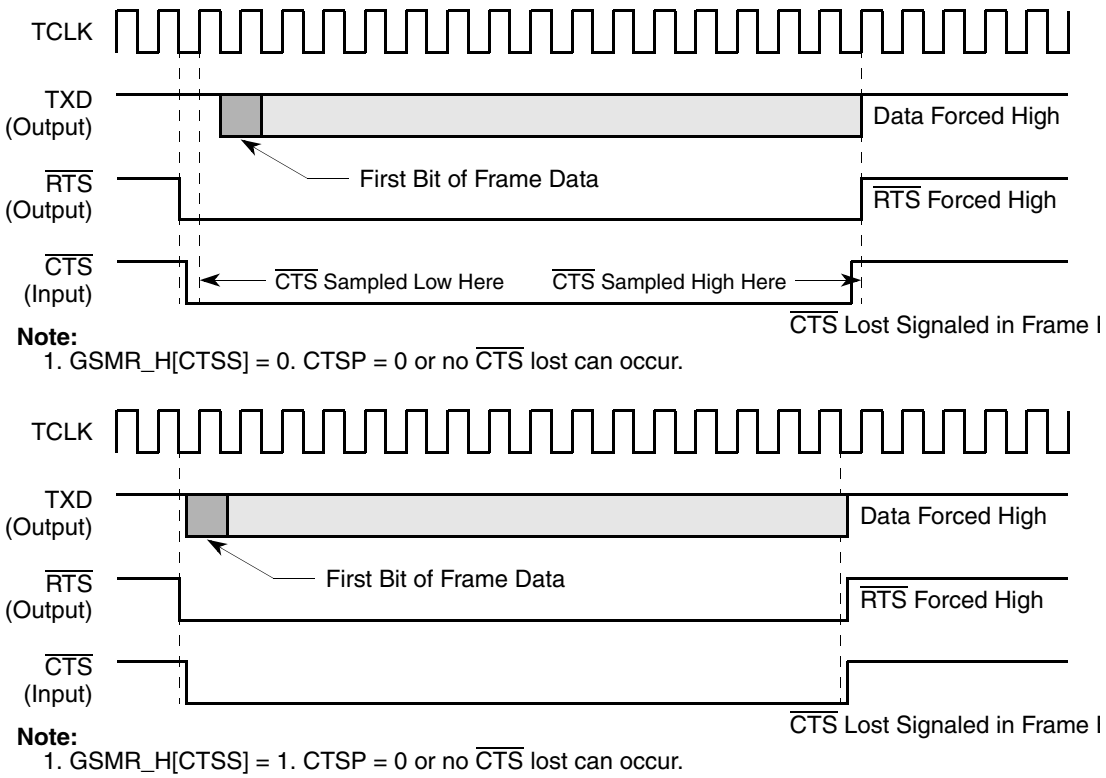


Figure 28-11. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

Note that if GSMR_H[CTSS] = 1, $\overline{\text{CTS}}$ transitions must occur while the Tx clock is low.

Reception delays are determined by $\overline{\text{CD}}$ as shown in Figure 28-12. If GSMR_H[CDS] is zero, $\overline{\text{CD}}$ is sampled on the rising Rx clock edge before data is received. If GSMR_H[CDS] is 1, $\overline{\text{CD}}$ transitions cause data to be immediately gated into the receiver.

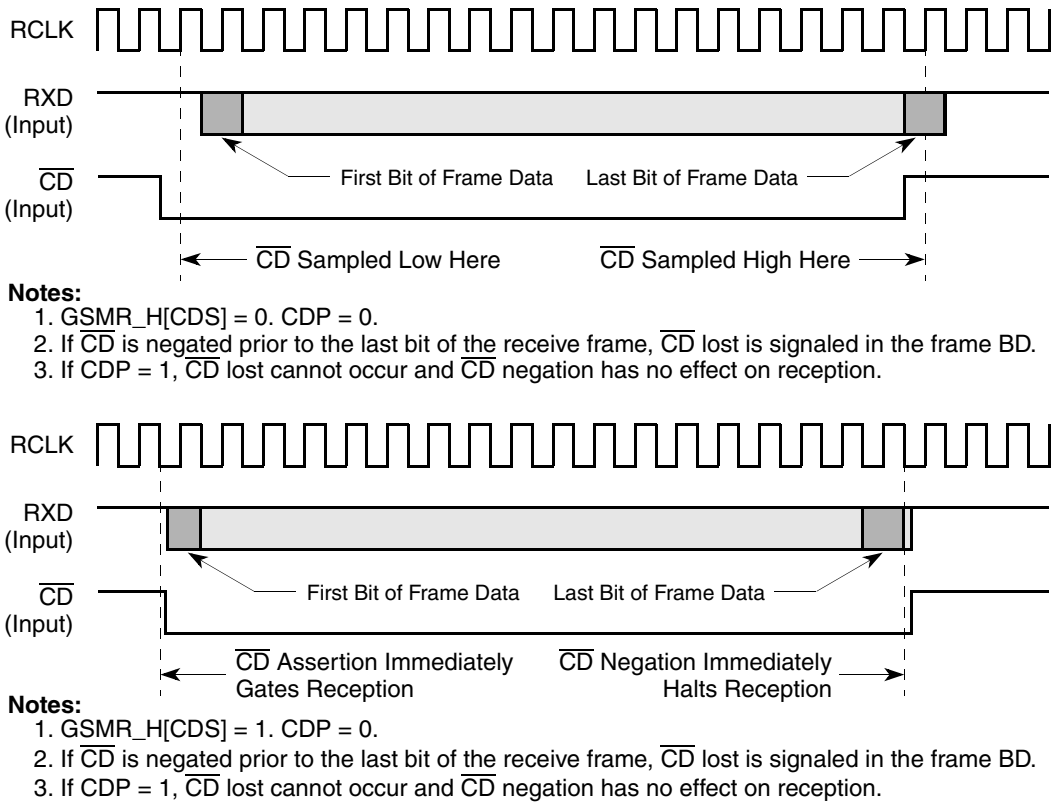


Figure 28-12. Using $\overline{CD}$ to Control Synchronous Protocol Reception

If $\overline{CD}$ is programmed to envelope the data, it must remain asserted during frame transmission or a $\overline{CD}$ lost error occurs. Negation of $\overline{CD}$ terminates reception. If GSMR_H[CDS] is zero, $\overline{CD}$ must be sampled by the SCC before a $\overline{CD}$ lost error is recognized; otherwise, the negation of $\overline{CD}$ immediately causes the $\overline{CD}$ lost condition.

If GSMR_H[CDS] is set, all $\overline{CD}$ transitions must occur while the Rx clock is low.

28.4.5.2 Asynchronous Protocols

In asynchronous protocols, $\overline{RTS}$ is asserted when SCC data is loaded into the Tx FIFO and a falling Tx clock occurs. $\overline{CD}$ and $\overline{CTS}$ can be used to control reception and transmission in the same manner as the synchronous protocols. The first bit sent in an asynchronous protocol is the start bit of the first character. In addition, the UART protocol has an option for $\overline{CTS}$ flow control as described in Chapter 29, "SCC UART Mode."

- If $\overline{CTS}$ is already asserted when $\overline{RTS}$ is asserted, transmission begins in two additional bit times.
- If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted and GSMR_H[CTSS] = 0, transmission begins in three additional bit times.
- If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted and GSMR_H[CTSS] = 1, transmission begins in two additional bit times.

28.4.6 Digital Phase-Locked Loop (DPLL) Operation

Each SCC channel includes a digital phase-locked loop (DPLL) for recovering clock information from a received data stream. For applications that provide a direct clock source to the SCC, the DPLL can be bypassed by selecting 1x mode for GSMR_L[RDCR, TDCR]. If the DPLL is bypassed, only NRZ or NRZI encodings are available. The DPLL is optional for protocols. Figure 28-13 shows the DPLL receiver block; Figure 28-14 shows the transmitter block diagram.

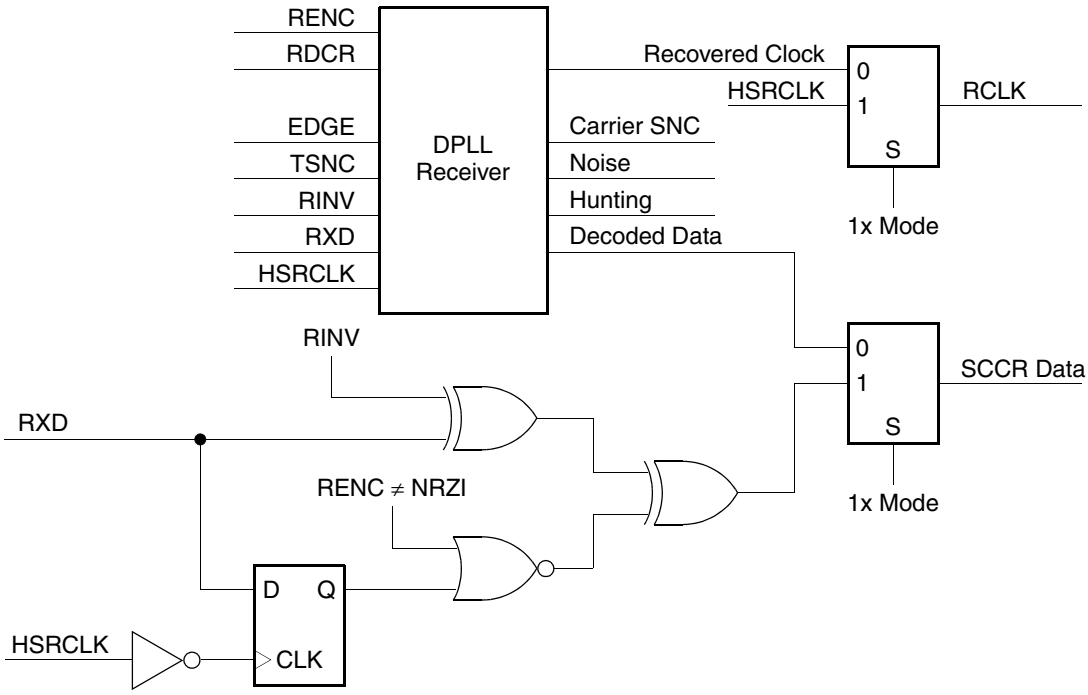


Figure 28-13. DPLL Receiver Block Diagram

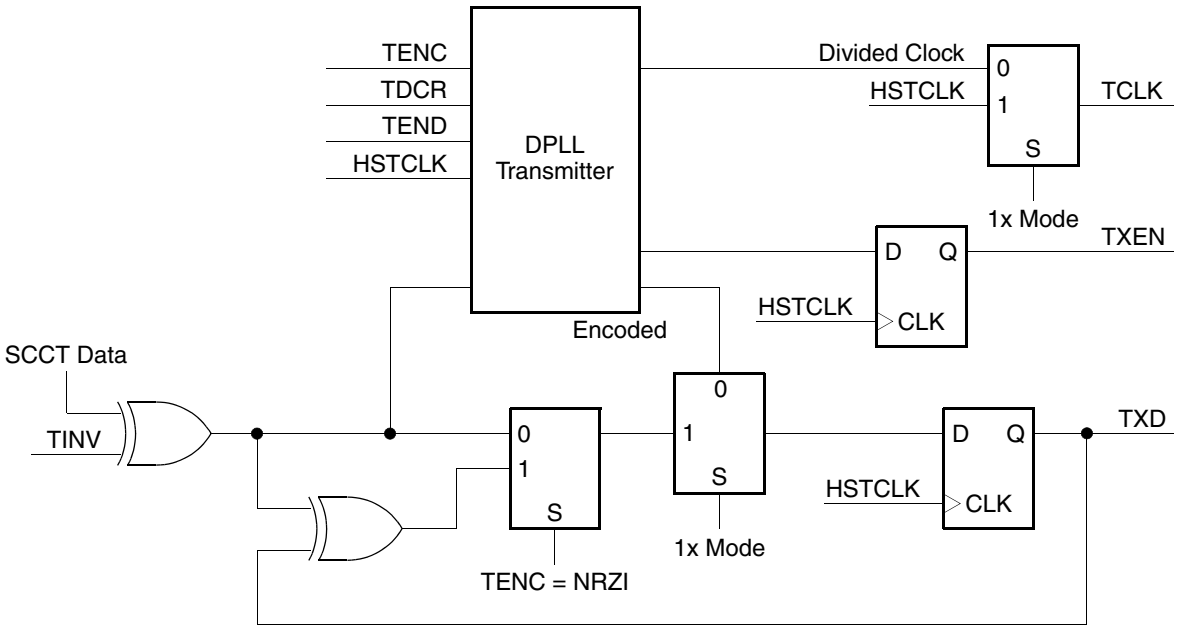


Figure 28-14. DPLL Transmitter Block Diagram

The DPLL can be driven by one of the baud rate generator outputs or an external clock, CLK*x*. In the block diagrams, this clock is labeled HSRCLK/HSTCLK. The HSRCLK/HSTCLK should be approximately 8x, 16x, or 32x the data rate, depending on the coding chosen. The DPLL uses this clock, along with the data stream, to construct a data clock that can be used as the SCC Rx and/or Tx clock. In all modes, the DPLL uses the input clock to determine the nominal bit time. If the DPLL is bypassed, HSRCLK/HSTCLK is used directly as RCLK/TCLK.

At the beginning of operation, the DPLL is in search mode, whereas the first transition resets the internal DPLL counter and begins DPLL operation. While the counter is counting, the DPLL watches the incoming data stream for transitions; when one is detected, the DPLL adjusts the count to produce an output clock that tracks incoming bits.

The DPLL has a carrier-sense signal that indicates when data transfers are on RXD. The carrier-sense signal asserts as soon as a transition is detected on RXD; it negates after the programmed number of clocks in GSMR_L[TSNC] when no transitions are detected.

To prevent itself from locking on the wrong edges and to provide fast synchronization, the DPLL should receive a preamble pattern before it receives the data. In some protocols, the preceding flags or syncs can function as a preamble; others use the patterns in Table 28-8. When transmission occurs, the SCC can generate preamble patterns, as programmed in GSMR_L[TPP, TPL].

Table 28-8. Preamble Requirements

Decoding Method	Preamble Pattern	Minimum Preamble Length Required
NRZI Mark	All zeros	8-bit
NRZI Space	All ones	8-bit
FM0	All ones	8-bit

Table 28-8. Preamble Requirements (continued)

Decoding Method	Preamble Pattern	Minimum Preamble Length Required
FM1	All zeros	8-bit
Manchester	101010...10	8-bit
Differential Manchester	All ones	8-bit

The DPLL can also be used to invert the data stream of a transfer. This feature is available in all encodings, including standard NRZ format. Also, when the transmitter is idling, the DPLL can either force TXD high or continue encoding the data supplied to it.

The DPLL is used for UART encoding/decoding, which gives the option of selecting the divide ratio in the UART decoding process (8×, 16×, or 32×). Typically, 16× is used.

Note the 1:4 system clock/serial clock ratio does not apply when the DPLL is used to recover the clock in the 8×, 16×, or 32× modes. Synchronization occurs internally after the DPLL generates the Rx clock. Therefore, even the fastest DPLL clock generation (the 8× option) easily meets the required 1:4 ratio clocking limit.

28.4.6.1 Encoding Data with a DPLL

Each SCC contains a DPLL unit that can be programmed to encode and decode the SCC data as NRZ, NRZI Mark, NRZI Space, FM0, FM1, Manchester, and Differential Manchester. Figure 28-15 shows the different encoding methods.

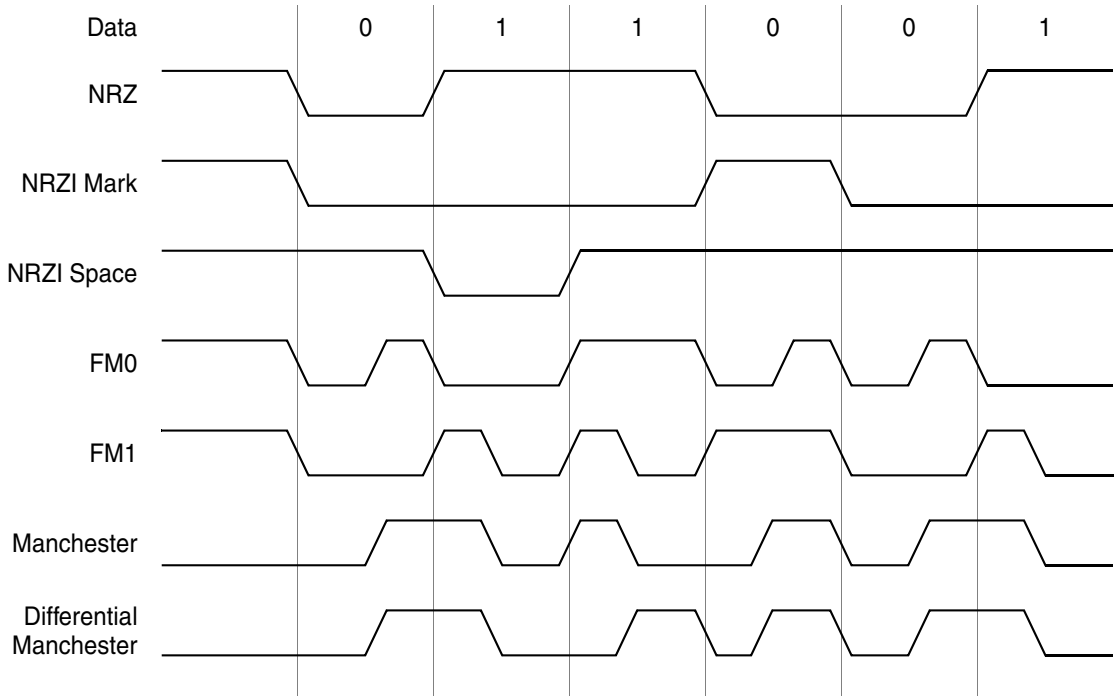


Figure 28-15. DPLL Encoding Examples

If the DPLL is not needed, NRZ or NRZI codings can be selected in GSMR_L[RENC, TENC]. Coding definitions are shown in Table 28-9.

Table 28-9. DPLL Codings

Coding	Description
NRZ	A one is represented by a high level for the duration of the bit and a zero is represented by a low level.
NRZI Mark	A one is represented by no transition at all. A zero is represented by a transition at the beginning of the bit (the level present in the preceding bit is reversed).
NRZI Space	A one is represented by a transition at the beginning of the bit (the level present in the preceding bit is reversed). A zero is represented by no transition at all.
FM0	A one is represented by a transition only at the beginning of the bit. A zero is represented by a transition at the beginning of the bit and another transition at the center of the bit.
FM1	A one is represented by a transition at the beginning of the bit and another transition at the center of the bit. A zero is represented by a transition only at the beginning of the bit.
Manchester	A one is represented by a high-to-low transition at the center of the bit. A zero is represented by a low to high transition at the center of the bit. In both cases there may be a transition at the beginning of the bit to set up the level required to make the correct center transition.
Differential Manchester	A one is represented by a transition at the center of the bit with the opposite direction from the transition at the center of the preceding bit. A zero is represented by a transition at the center of the bit with the same polarity from the transition at the center of the preceding bit.

28.4.7 Reconfiguring the SCCs

The proper reconfiguration sequence must be followed for SCC parameters that cannot be changed dynamically. For instance, the internal baud rate generators allow on-the-fly changes, but the DPLL-related GSMR does not. The steps in the following sections show how to disable, reconfigure and re-enable an SCC to ensure that buffers currently in use are properly closed before reconfiguring the SCC and that subsequent data goes to or from new buffers according to the new configuration.

Modifying parameter RAM does not require the SCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CPCR[RST] to reset the entire CPM.

28.4.7.1 General Reconfiguration Sequence for an SCC Transmitter

An SCC transmitter can be reconfigured by following these general steps:

1. If the SCC is sending data, issue a STOP TRANSMIT command. Transmission should stop smoothly. If the SCC is not transmitting (no TxBDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and completed) or the INIT TX PARAMETERS command is issued, the STOP TRANSMIT command is not required.
2. Clear GSMR_L[ENT] to disable the SCC transmitter and put it in reset state.
3. Modify SCC Tx parameters or parameter RAM. To switch protocols or restore the initial Tx parameters, issue an INIT TX PARAMETERS command.
4. If an INIT TX PARAMETERS command was not issued in step 3, issue a RESTART TRANSMIT command.

5. Set GSMR_L[ENT]. Transmission begins using the TxBD pointed to by TBPTR, assuming the R bit is set.

28.4.7.2 Reset Sequence for an SCC Transmitter

The following steps reinitialize an SCC transmit parameters to the reset state:

1. Clear GSMR_L[ENT].
2. Make any modifications then issue the INIT TX PARAMETERS command.
3. Set GSMR_L[ENT].

28.4.7.3 General Reconfiguration Sequence for an SCC Receiver

An SCC receiver can be reconfigured by following these steps:

1. Clear GSMR_L[ENR]. The SCC receiver is now disabled and put in a reset state.
2. Modify SCC Rx parameters or parameter RAM. To switch protocols or restore Rx parameters to their initial state, issue an INIT RX PARAMETERS command.
3. If the INIT RX PARAMETERS command was not issued in step 2, issue an ENTER HUNT MODE command.
4. Set GSMR_L[ENR]. Reception begins using the RxBD pointed to by RBPTR, assuming the E bit is set.

28.4.7.4 Reset Sequence for an SCC Receiver

To reinitialize the SCC receiver to the state it was in after reset, follow these steps:

1. Clear GSMR_L[ENR].
2. Make any modifications then issue the INIT RX PARAMETERS command.
3. Set GSMR_L[ENR].

28.4.7.5 Switching Protocols

To switch an SCC's protocol without resetting the board or affecting other SCCs, follow these steps:

1. Clear GSMR_L[ENT, ENR].
2. Make protocol changes in the GSMR and additional parameters then issue the INIT TX and RX PARAMETERS command to initialize both Tx and Rx parameters.
3. Set GSMR_ L[ENT, ENR] to enable the SCC with the new protocol.

28.4.8 Saving Power

To save power when not in use, an SCC can be disabled by clearing GSMR_L[ENT, ENR].

Chapter 29
SCC UART Mode

The universal asynchronous receiver transmitter (UART) protocol is commonly used to send low-speed data between devices. The term asynchronous is used because it is not necessary to send clocking information along with the data being sent. UART links are typically 38,400 baud or less and are character-based. Asynchronous links are used to connect terminals with other devices. Even where synchronous communications are required, the UART is often used as a local port to run board debugger software. The character format of the UART protocol is shown in Figure 29-1.

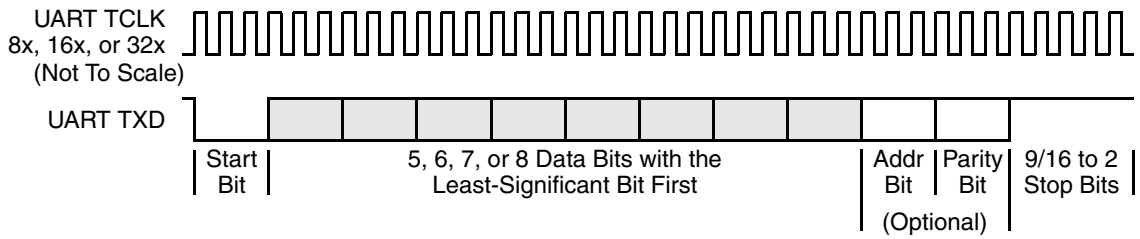


Figure 29-1. UART Character Format

Because the transmitter and receiver operate asynchronously, there is no need to connect the transmit and receive clocks. Instead, the receiver oversamples the incoming data stream (usually by a factor of 16) and uses some of these samples to determine the bit value. Traditionally, the middle 3 of the 16 samples are used. Two UARTs can communicate using this system if the transmitter and receiver use the same parameters, such as the parity scheme and character length.

When data is not sent, a continuous stream of ones is sent (idle condition). Because the start bit is always a zero, the receiver can detect when real data is once again on the line. UART specifies an all-zeros break character, which ends a character transfer sequence.

The most popular protocol that uses asynchronous characters is the RS-232 standard, which specifies baud rates, handshaking protocols, and mechanical/electrical details. Another popular format is RS-485, which defines a balanced line system allowing longer cables than RS-232 links. Even synchronous protocols like HDLC are sometimes defined to run over asynchronous links. The Profibus standard extends UART protocol to include LAN-oriented features such as token passing.

All standards provide handshaking signals, but some systems require only three physical lines—Tx data, Rx data, and ground. Many proprietary standards have been built around the UART's asynchronous character frame, some of which implement a multidrop configuration where multiple stations, each with a specific address, can be present on a network. In multidrop mode, frames of characters are broadcast with the first character acting as a destination address. To accommodate this, the UART frame is extended one bit to distinguish address characters from normal data characters.

In synchronous UART (isochronous operation), a separate clock signal is explicitly provided with the data. Start and stop bits are present in synchronous UART, but oversampling is not required because the clock is provided with each bit.

The general SCC mode register (GSMR) is used to configure an SCC channel to function in UART mode, which provides standard serial I/O using asynchronous character-based (start-stop) protocols with RS-232C-type lines. Using standard asynchronous bit rates and protocols, an SCC UART controller can communicate with any existing RS-232-type device and provides a serial communications port to other microprocessors and terminals (either locally or through modems). The independent transmit and receive sections, whose operations are asynchronous with the core, send data from memory (either internal or external) to TXD and receive data from RXD. The UART controller supports a multidrop mode for master/slave operations with wakeup capability on both the idle signal and address bit. It also supports synchronous operation where a clock (internal or external) must be provided with each bit received.

29.1 Features

The following list summarizes main features of an SCC UART controller:

- Flexible message-based data structure
- Implements synchronous and asynchronous UART
- Multidrop operation
- Receiver wake-up on idle line or address bit
- Receive entire messages into buffers as indicated by receiver idle timeout or by control character reception
- Eight control character comparison
- Two address comparison in multidrop configurations
- Maintenance of four 16-bit error counters
- Received break character length indication
- Programmable data length (5–8 bits)
- Programmable fractional stop bit lengths (from 9/16 to 2 bits) in transmission
- Capable of reception without a stop bit
- Even/odd/force/no parity generation and check
- Frame error, noise error, break, and idle detection
- Transmit preamble and break sequences
- Freeze transmission option with low-latency stop

29.2 Normal Asynchronous Mode

In normal asynchronous mode, the receive shift register receives incoming data on RXD*x*. Control bits in the UART mode register (PSMR) define the length and format of the UART character. Bits are received in the following order:

1. Start bit
2. 5–8 data bits (lsb first)

3. Address/data bit (optional)
4. Parity bit (optional)
5. Stop bits

The receiver uses a clock 8x 16x, or 32x faster than the baud rate and samples each bit of the incoming data three times around its center. The value of the bit is determined by the majority of those samples; if all do not agree, the noise indication counter (NOSEC) in parameter RAM is incremented. When a complete character has been clocked in, the contents of the receive shift register are transferred to the receive FIFO before proceeding to the receive buffer. The CPM flags UART events, including reception errors, in SCCE and the RxBD status and control fields.

The SCC can receive fractional stop bits. The next character's start bit can begin any time after the three middle samples are taken. The UART transmit shift register sends outgoing data on TXD*x*. Data is then clocked synchronously with the transmit clock, which may have either an internal or external source. Characters are sent lsb first. Only the data portion of the UART frame is stored in the buffers because start and stop bits are generated and stripped by the SCC. A parity bit can be generated in transmission and checked during reception; although it is not stored in the buffer, its value can be inferred from the buffer's reporting mechanism. Similarly, the optional address bit is not stored in the transmit or receive buffer, but is supplied in the BD itself. Parity generation and checking includes the optional address bit. GSMR_H[RFW] must be set for an 8-bit receive FIFO in the UART receiver.

29.3 Synchronous Mode

In synchronous mode, the controller uses a 1x data clock for timing. The receive shift register receives incoming data on RXD*x* synchronous with the clock. The bit length and format of the serial character are defined by the control bits in the PSMR in the same way as in asynchronous mode. When a complete byte has been clocked in, the contents of the receive shift register are transferred to the receive FIFO before proceeding to the receive buffer. The CPM flags UART events, including reception errors, in SCCE and the RxBD status and control fields. GSMR_H[RFW] must be set for an 8-bit receive FIFO.

The synchronous UART transmit shift register sends outgoing data on TXD*x*. Data is then clocked synchronously with the transmit clock, which can have an internal or external source.

29.4 SCC UART Parameter RAM

For UART mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 29-1.

Table 29-1. UART-Specific SCC Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x30	—	DWord	Reserved
0x38	MAX_IDL	Hword	Maximum idle characters. When a character is received, the receiver begins counting idle characters. If MAX_IDL idle characters are received before the next data character, an idle timeout occurs and the buffer is closed, generating a maskable interrupt request to the core to receive the data from the buffer. Thus, MAX_IDL offers a way to demarcate frames. To disable the feature, clear MAX_IDL. The bit length of an idle character is calculated as follows: 1 + data length (5–9) + 1 (if parity is used) + number of stop bits (1–2). For 8 data bits, no parity, and 1 stop bit, the character length is 10 bits.

Table 29-1. UART-Specific SCC Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
0x3A	IDLC	Hword	Temporary idle counter. Holds the current idle count for the idle timeout process. IDLC is a down-counter and does not need to be initialized or accessed.
0x3C	BRKCR	Hword	Break count register (transmit). Determines the number of break characters the transmitter sends. The transmitter sends a break character sequence when a STOP TRANSMIT command is issued. For 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character consists of 10 zero bits.
0x3E	PAREC	Hword	User-initialized,16-bit (modulo–2^{16}) counters incremented by the CP. PAREC counts received parity errors. FRMEC counts received characters with framing errors. NOSEC counts received characters with noise errors. BRKEC counts break conditions on the signal. A break condition can last for hundreds of bit times, yet BRKEC is incremented only once during that period.
0x46	BRKLN	Hword	Last received break length. Holds the length of the last received break character sequence measured in character units. For example, if RXD*x* is low for 20 bit times and the defined character length is 10 bits, BRKLN = 0 × 002, indicating that the break sequence is at least 2 characters long. BRKLN is accurate to within one character length.
0x48	UADDR1	Hword	UART address character 1/2. In multidrop mode, the receiver provides automatic address recognition for two addresses. In this case, program the lower order bytes of UADDR1 and UADDR2 with the two preferred addresses.
0x4A	UADDR2	Hword	
0x50	CHARACTER1	Hword	Control character 1–8. These characters define the Rx control characters on which interrupts can be generated.
0x52	CHARACTER2	Hword	
0x54	CHARACTER3	Hword	
0x56	CHARACTER4	Hword	
0x58	CHARACTER5	Hword	
0x5A	CHARACTER6	Hword	
0x5C	CHARACTER7	Hword	
0x5E	CHARACTER8	Hword	
0x60	RCCM	Hword	Receive control character mask. Used to mask comparison of CHARACTER1–8 so classes of control characters can be defined. A one enables the comparison, and a zero masks it.
0x62	RCCR	Hword	Receive control character register. Used to hold the last rejected control character (not written to the Rx buffer). Generates a maskable interrupt. If the core does not process the interrupt and read RCCR before a new control character arrives, the previous control character is overwritten.
0x64	RLBC	Hword	Receive last break character. Used in synchronous UART when PSMR[RZS] = 1; holds the last break character pattern. By counting zeros in RLBC, the core can measure break length to a one-bit resolution. Read RLBC by counting the zeros written from bit 0 to where the first one was written. RLBC = 0b001xxxxxxxxxxxxx indicates two zeros; 0b1xxxxxxxxxxxxxxx indicates no zeros. Note that RLBC can be used in combination with BRKLN above to calculate the number of bits in the break sequence: (BRKLN × character length) + (number of zeros in RLBC).

[1] From SCC base. See Section 28.4.1, "SCC Base Addresses."

29.5 Data-Handling Methods: Character- or Message-Based

An SCC UART controller uses the same BD table and buffer structures as the other protocols and supports both multi buffer, message-based and single-buffer, character-based operation.

For character-based transfers, each character is sent with stop bits and parity and received into separate 1-byte buffers. A maskable interrupt is generated when each buffer is received.

In a message-based environment, transfers can be made on entire messages rather than on individual characters. To simplify programming and save processor overhead, a message is transferred as a linked list of buffers without core intervention. For example, before handling input data, a terminal driver may wait for an end-of-line character or an idle timeout rather than be interrupted when each character is received. Conversely, ASCII files can be sent as messages ending with an end-of-line character.

When receiving messages, up to eight control characters can be configured to mark the end of a message or generate a maskable interrupt without being stored in the buffer. This option is useful when flow control characters such as XON or XOFF are needed but are not part of the received message. See Section 29.9, “Receiving Control Characters.”

29.6 Error and Status Reporting

Overrun, parity, noise, and framing errors are reported through the BDs and/or error counters in the UART parameter RAM. Signal status is indicated in the status register; a maskable interrupt is generated when status changes.

29.7 SCC UART Commands

The transmit commands in Table 29-2 are issued to the CP command register (CPCR).

Table 29-2. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GSMR, the transmitter starts polling the first BD in the TxBD table every 8 Tx clocks. STOP TRANSMIT disables character transmission. If the SCC receives STOP TRANSMIT as a message is being sent, the message is aborted. The transmitter finishes sending data transferred to its FIFO and stops. The TBPTR is not advanced. The UART transmitter sends a programmable break sequence and starts sending idles. The number of break characters in the sequence (which can be zero) should be written to BRKCR in the parameter RAM before issuing this command.
GRACEFUL STOP TRANSMIT	Used to stop transmitting smoothly. The transmitter stops after the current buffer has been completely sent or immediately if no buffer is being sent. SCCE[GRA] is set once transmission stops, then the UART Tx parameters, including the TxBD, can be modified. TBPTR points to the next TxBD in the table. Transmission begins once the R bit of the next BD is set and a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Enables transmission. The controller expects this command after it disables the channel in its PSMR, after a STOP TRANSMIT command, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error. Transmission resumes from the current BD.
INIT TX PARAMETERS	Resets the transmit parameters in the parameter RAM. Issue only when the transmitter is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

Receive commands are described in Table 29-3.

Table 29-3. Receive Commands

Command	Description
ENTER HUNT MODE	Forces the receiver to close the RxBD in use and enter hunt mode. After a hardware or software reset, once an SCC is enabled in the GSMR, the receiver is automatically enabled and uses the first BD in the RxBD table. If a message is in progress, the receiver continues receiving in the next BD. In multidrop hunt mode, the receiver continually scans the input data stream for the address character. When it is not in multidrop mode, it waits for the idle sequence (one character of idle). Data present in the Rx FIFO is not lost when this command is executed.
CLOSE RXBD	Forces the SCC to close the RxBD in use and use the next BD for subsequent received data. If the SCC is not in the process of receiving data, no action is taken. Note that in an SCC UART controller, CLOSE RXBD functions like ENTER HUNT MODE but does not need to receive an idle character to continue receiving.
INIT RX PARAMETERS	Resets the receive parameters in the parameter RAM. Should be issued when the receiver is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

29.8 Multidrop Systems and Address Recognition

In multidrop systems, more than two stations can be on a network, each with a specific address. Figure 29-2 shows two examples of this configuration. Frames made up of many characters can be broadcast as long as the first character is the destination address. The UART frame is extended by one bit to distinguish an address character from standard data characters. Programmed in PSMR[UM], the controller supports the following two multidrop modes:

- Automatic multidrop mode—The controller checks the incoming address character and accepts subsequent data only if the address matches one of two user-defined values. The two 16-bit address registers, UADDR1 and UADDR2, support address recognition. Only the lower 8 bits are used so the upper 8 bits should be cleared; for addresses less than 8 bits, unused high-order bits should also be cleared. The incoming address is checked against UADDR1 and UADDR2. When a match occurs, RxBD[AM] indicates whether UADDR1 or UADDR2 matched.
- Manual multidrop mode—The controller receives all characters. An address character is always written to a new buffer and can be followed by data characters. User software performs the address comparison.

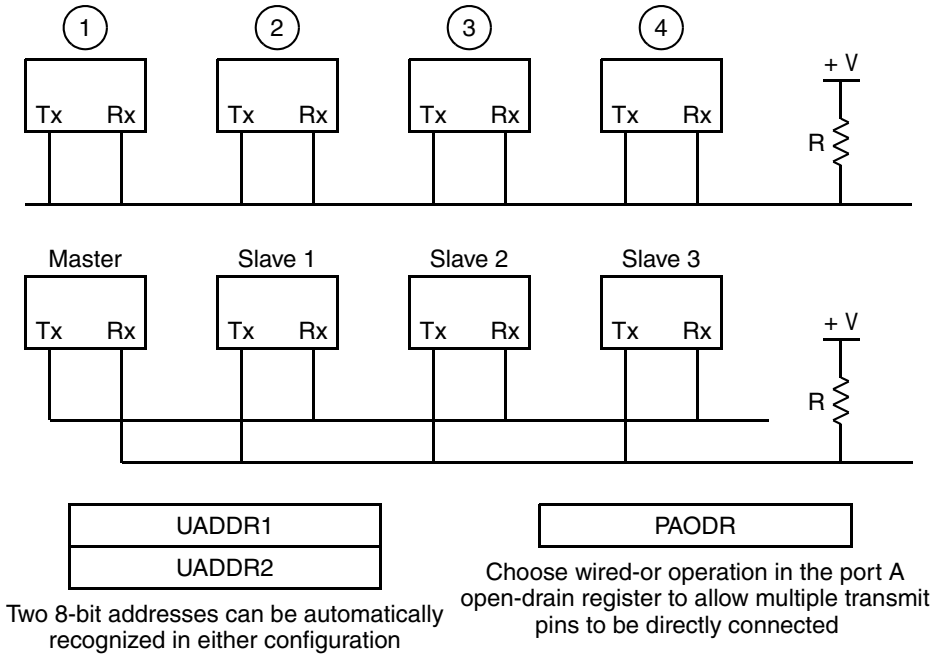


Figure 29-2. Two UART Multidrop Configurations

29.9 Receiving Control Characters

The UART receiver can recognize special control characters used in a message-based environment. Eight control characters can be defined in a control character table in the UART parameter RAM. Each incoming character is compared to the table entries using a mask (the received control character mask, RCCM) to strip don't cares. If a match occurs, the received control character can either be written to the receive buffer or rejected.

If the received control character is not rejected, it is written to the receive buffer. The receive buffer is then automatically closed to allow software to handle end-of-message characters. Control characters that are not part of the actual message, such as XOFF, can be rejected. Rejected characters bypass the receive buffer and are written directly to the received control character register (RCCR), which triggers maskable interrupt.

The 16-bit entries in the control character table support control character recognition. Each entry consists of the control character, a valid bit (end of table), and a reject bit. See Figure 29-3.

Offset [1]	0	1	2–7	8–15
0x50	E	R	—	CHARACTER1
0x52	E	R	—	CHARACTER2
⋮	⋮	⋮	⋮	⋮
0x5E	E	R	—	CHARACTER8
0x60	1	1	—	RCCM
0x62	—			RCCR

[1] From SCC*x* base address.

Figure 29-3. Control Character Table

Table 29-4 describes the data structure used in control character recognition.

Table 29-4. Control Character Table, RCCM, and RCCR Descriptions

Offset	Bits	Name	Description
0x50–0x5E	0	E	End of table. In tables with eight control characters, E is always 0. 0 This entry is valid. 1 The entry is not valid and is not used.
	1	R	Reject character. 0 A matching character is not rejected but is written into the Rx buffer, which is then closed. If RxBD[I] is set, the buffer closing generates a maskable interrupt through SCCE[RX]. A new buffer is opened if more data is in the message. 1 A matching character is written to RCCR and not to the Rx buffer. A maskable interrupt is generated through SCCE[CCR]. The current Rx buffer is not closed.
	2–7	—	Reserved
	8–15	CHARACTERn	Control character values 1–8. Defines control characters to be compared to the incoming character. For characters smaller than 8 bits, the most significant bits should be zero.
0x60	0–1	0b11	Must be set. Used to mark the end of the control character table in case eight characters are used. Setting these bits ensures correct operation during control character recognition.
	2–7	—	Reserved
	8–15	RCCM	Received control character mask. Used to mask the comparison of CHARACTER*n*. Each RCCM bit corresponds to the respective bit of CHARACTER*n* and decodes as follows. 0 Ignore this bit when comparing the incoming character to CHARACTER*n*. 1 Use this bit when comparing the incoming character to CHARACTER*n*.
0x62	0–7	—	Reserved
	8–15	RCCR	Received control character register. If the newly arrived character matches and is rejected from the buffer (R = 1), the PIP controller writes the character into the RCCR and generates a maskable interrupt. If the core does not process the interrupt and read RCCR before a new control character arrives, the previous control character is overwritten.

29.10 Hunt Mode (Receiver)

A UART receiver in hunt mode remains deactivated until an idle or address character is recognized, depending on PSMR[UM]. A receiver is forced into hunt mode by issuing an ENTER HUNT MODE command.

The receiver aborts any message in progress when ENTER HUNT MODE is issued. When the message is finished, the receiver is re-enabled by detecting the idle line (one idle character) or by the address bit of the next message, depending on PSMR[UM]. When a receiver in hunt mode receives a break sequence, it increments BRKEC and generates a BRK interrupt condition.

29.11 Inserting Control Characters into the Transmit Data Stream

The SCC UART transmitter can send out-of-sequence, flow-control characters like XON and XOFF. The controller polls the transmit out-of-sequence register (TOSEQ), shown in Figure 29-4, whenever the transmitter is enabled for UART operation, including during a UART freeze operation, UART buffer transmission, and when no buffer is ready for transmission. The TOSEQ character (in CHARSEND) is sent at a higher priority than the other characters in the transmit buffer, but does not preempt characters already in the transmit FIFO. This means that the XON or XOFF character may not be sent for eight or four (SCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

	0–1	2	3	4	5–6	7	8–15
Field	—	REA	I	CT	—	A	CHARSEND
Reset	0000_0000_0000_0000						
R/W	R/W						
Offset	SCC base + 0x4E						

Figure 29-4. Transmit Out-of-Sequence Register (TOSEQ)

Table 29-5 describes TOSEQ fields.

Table 29-5. TOSEQ Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	REA	Ready. Set when the character is ready for transmission. Remains 1 while the character is being sent. The CP clears this bit after transmission.
3	I	Interrupt. If this bit is set, transmission completion is flagged in the event register (SCCE[TX] is set), triggering a maskable interrupt to the core.
4	CT	Clear-to-send lost. Operates only if the SCC monitors $\overline{\text{CTS}}$ (GSMR_L[DIAG]). The CP sets this bit if $\overline{\text{CTS}}$ negates when the TOSEQ character is sent. If $\overline{\text{CTS}}$ negates and the TOSEQ character is sent during a buffer transmission, the TxBD[CT] status bit is also set.
5–6	—	Reserved, should be cleared.

Table 29-5. TOSEQ Field Descriptions (continued)

Bits	Name	Description
7	A	Address. Setting this bit indicates an address character for multidrop mode.
8–15	CHARSEND	Character send. Contains the character to be sent. Any 5- to 8-bit character value can be sent in accordance with the UART configuration. The character should be placed in the lsbs of CHARSEND. This value can be changed only while REA = 0.

29.12 Sending a Break (Transmitter)

A break is an all-zeros character with no stop bit that is sent by issuing a STOP TRANSMIT command. The SCC finishes transmitting outstanding data, sends a programmable number of break characters (determined by BRKCR), and reverts to idle or sends data if a RESTART TRANSMIT command is given before completion. When the break code is complete, the transmitter sends at least one high bit before sending more data, to guarantee recognition of a valid start bit. Because break characters do not preempt characters in the transmit FIFO, they may not be sent for eight (SCC) or four (SCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

29.13 Sending a Preamble (Transmitter)

Sending a preamble sequence of consecutive ones ensures that a line is idle before sending a message. If the preamble bit TxBD[P] is set, the SCC sends a preamble sequence (idle character) before sending the buffer. For example, for 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of 10 ones is sent before the first character in the buffer.

29.14 Fractional Stop Bits (Transmitter)

The asynchronous UART transmitter, shown in Figure 29-5, can be programmed to send fractional stop bits. The FSB field in the data synchronization register (DSR) determines the fractional length of the last stop bit to be sent. FSB can be modified at any time. If two stop bits are sent, only the second is affected. Idle characters are always sent as full-length characters.

	0	1 4	5	6	7	8	9	10	11	12	13	14	15
Field	—	FSB	—	—	—	—	—	—	—	—	—	—	—
Reset	0	1111	1	1	0	0	1	1	1	1	1	1	0
R/W	R/W												
Offset													

Figure 29-5. Asynchronous UART Transmitter

Table 29-6 describes DSR fields.

Table 29-6. DSR Fields Descriptions

Bits	Name	Description
0	—	0b0
1–4	FSB	Fractional stop bits. For 16× oversampling: 1111 Last transmitted stop bit 16/16. Default value after reset. 1110 Last transmitted stop bit 15/16 …. 1000 Last transmitted stop bit 9/16. 0xxx Invalid. Do not use. For 32× oversampling: 1111 Last transmitted stop bit 32/32. Default value after reset. 1110 Last transmitted stop bit 31/32. …. 0000 Last transmitted stop bit 17/32. For 8× oversampling: 1111 Last transmitted stop bit 8/8. Default value after reset. 1110 Last transmitted stop bit 7/8. 1101 Last transmitted stop bit 6/8. 1100 Last transmitted stop bit 5/8. 10xx Invalid. Do not use. 0xxx Invalid. Do not use. The UART receiver can always receive fractional stop bits. The next character's start bit can begin any time after the three middle samples have been taken.
5–6	—	0b11
7–8	—	0b00
9–14	—	0b111111
15	—	0b0

29.15 Handling Errors in the SCC UART Controller

The UART controller reports character reception and transmission error conditions through the BDs, the error counters, and the SCCE. Modem interface lines can be monitored by the port C pins. Transmission errors are described in Table 29-7.

Table 29-7. Transmission Errors

Error	Description
$\overline{\text{CTS}}$ Lost during Character Transmission	When $\overline{\text{CTS}}$ negates during transmission, the channel stops after finishing the current character. The CP sets TxBD[CT] and generates the TX interrupt if it is not masked. The channel resumes transmission after the RESTART TRANSMIT command is issued and $\overline{\text{CTS}}$ is asserted. Note that if $\overline{\text{CTS}}$ is used, the UART also offers an asynchronous flow control option that does not generate an error. See the description of PSMR[FLC] in Table 29-9.

Reception errors are described in Table 29-8.

Table 29-8. Reception Errors

Error	Description
Overrun	Occurs when the channel overwrites the previous character in the Rx FIFO with a new character, losing the previous character. The channel then writes the new character to the buffer, closes it, sets RxBD[OV], and generates an RX interrupt if not masked. In automatic multidrop mode, the receiver enters hunt mode immediately.
$\overline{\text{CD}}$ Lost during Character Reception	If this error occurs and the channel is using this pin to automatically control reception, the channel terminates character reception, closes the buffer, sets RxBD[CD], and generates the RX interrupt if not masked. This error has the highest priority. The last character in the buffer is lost and other errors are not checked. In automatic multidrop mode, the receiver enters the hunt mode immediately.
Parity	When a parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets RxBD[PR], and generates the RX interrupt if not masked. The channel also increments the parity error counter PAREC. In automatic multidrop mode, the receiver enters hunt mode immediately.
Noise	A noise error occurs when the three samples of a bit are not identical. When this error occurs, the channel writes the received character to the buffer, proceeds normally, but increments the noise error counter NOSEC. Note that this error does not occur in synchronous mode.
Idle Sequence Receive	If the UART is receiving data and gets an idle character (all ones), the channel begins counting consecutive idle characters received. If MAX_IDL is reached, the buffer is closed and an RX interrupt is generated if not masked. If no buffer is open, this event does not generate an interrupt or any status information. The internal idle counter (IDLC) is reset every time a character is received. To disable the idle sequence function, clear MAX_IDL.
Framing	The UART reports a framing errors when it receives a character with no stop bit, regardless of the mode. The channel writes the received character to the buffer, closes it, sets RxBD[FR], generates the RX interrupt if not masked, increments FRMEC, but does not check parity for this character. In automatic multidrop mode, the receiver immediately enters hunt mode. If the UART allows data with no stop bits (PSMR[RZS] = 1) when in synchronous mode (PSMR[SYN] = 1), framing errors are reported but reception continues assuming the unexpected zero is the start bit of the next character; in this case, the user may ignore a reported framing error until multiple framing errors occur within a short period.
Break Sequence	When the first break sequence is received, the UART increments the break error counter BRKEC. It updates BRKLN when the sequence completes. After the first 1 is received, the UART sets SCCE[BRKE], which generates an interrupt if not masked. If the UART is receiving characters when it receives a break, it closes the Rx buffer, sets RxBD[BR], and sets SCCE[RX], which can generate an interrupt if not masked. If PSMR[RZS] = 1 when the UART is in synchronous mode, a break sequence is detected after two successive break characters are received.

29.16 UART Mode Register (PSMR)

For UART mode, the SCC protocol-specific mode register (PSMR) is called the UART mode register. Many bits can be modified while the receiver and transmitter are enabled. Figure 29-6 shows the PSMR in UART mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FLC	SL	CL		UM		FRZ	RZS	SYN	DRT	—	PEN	RPM		TPM	
Reset	0															
R/W	R/W															
Offset	0x9_1A08 (PSMR1); 0x9_1A48 (PSMR3); 0x9_1A68 (PSMR4)															

Figure 29-6. Protocol-Specific Mode Register for UART (PSMR)

Table 29-9 describes PSMR UART fields.

Table 29-9. PSMR UART Field Descriptions

Bits	Name	Description
0	FLC	Flow control 0 Normal operation. The GSMR and port C registers determine the mode of $\overline{\text{CTS}}$. 1 Asynchronous flow control. When $\overline{\text{CTS}}$ is negated, the transmitter stops at the end of the current character. If $\overline{\text{CTS}}$ is negated past the middle of the current character, the next full character is sent before transmission stops. When $\overline{\text{CTS}}$ is asserted again, transmission continues where it left off and no $\overline{\text{CTS}}$ lost error is reported. Only idle characters are sent while $\overline{\text{CTS}}$ is negated.
1	SL	Stop length. Selects the number of stop bits the SCC sends. SL can be modified on-the-fly. The receiver is always enabled for one stop bit unless the SCC UART is in synchronous mode and PSMR[RZS] is set. Fractional stop bits are configured in the DSR. 0 1stop bit 1 2 stop bits
2–3	CL	Character length. Determines the number of data bits in the character, not including optional parity or multidrop address bits. If a character is less than 8 bits, most-significant bits are received as zeros and are ignored when the character is sent. CL can be modified on-the-fly. 00 5 data bits 01 6 data bits 10 7 data bits 11 8 data bits
4–5	UM	UART mode. Selects the asynchronous channel protocol. UM can be modified on-the-fly. 00 Normal UART operation. Multidrop mode is disabled and idle-line wake-up mode is selected. The UART receiver leaves hunt mode by receiving an idle character (all ones). 01 Manual multidrop mode. An additional address/data bit is sent with each character. Multidrop asynchronous modes are compatible with the MC68681 DUART, MC68HC11 SCI, DSP56000 SCI, and Intel 8051 serial interface. The receiver leaves hunt mode when the address/data bit is a one, indicating the received character is an address that all inactive processors must process. The controller receives the address character and writes it to a new buffer. The core then compares the written address with its own address and decides whether to ignore or process subsequent characters. 10 Reserved. 11 Automatic multidrop mode. The CPM compares the address of an incoming address character with UADDR*x* parameter RAM values; subsequent data is accepted only if a match occurs.
6	FRZ	Freeze transmission. Allows the UART transmitter to pause and later continue from that point. 0 Normal operation. If the buffer was previously frozen, it resumes transmission from the next character in the same buffer that was frozen. 1 The SCC completes transmission of any data already transferred to the Tx FIFO (the number of characters depends on GSMR_H[TFL]) and then freezes. After FRZ is cleared, transmission resumes from the next character.

Table 29-9. PSMR UART Field Descriptions (continued)

Bits	Name	Description
7	RZS	Receive zero stop bits 0 The receiver operates normally, but at least one stop bit is needed between characters. A framing error is issued if a stop bit is missing. Break status is set if an all-zero character is received with a zero stop bit. 1 Configures the receiver to receive data without stop bits. Useful in V.14 applications where SCC UART controller data is supplied synchronously and all stop bits of a particular character can be omitted for cross-network rate adaptation. RZS should be set only if SYN is set. The receiver continues if a stop bit is missing. If the stop bit is a zero, the next bit is considered the first data bit of the next character. A framing error is issued if a stop bit is missing, but a break status is reported only after two consecutive break characters have no stop bits.
8	SYN	Synchronous mode 0 Normal asynchronous operation. GSMR_L[TENC,RENC] must select NRZ and GSMR_L[TDCR, RDCR] select either 8×, 16×, or 32×. 16× is recommended for most applications. 1 Synchronous SCC UART controller using 1× clock (isochronous UART operation). GSMR_L[TENC, RENC] must select NRZ and GSMR_L[RDCR, TDCR] select 1× mode. A bit is transferred with each clock and is synchronous to the clock, which can be internal or external.
9	DRT	Disable receiver while transmitting 0 Normal operation 1 While the SCC is sending data, the internal $\overline{\text{RTS}}$ disables and gates the receiver. Useful for a multidrop configuration in which the user does not want to receive its own transmission. For multidrop UART mode, set the BDs' preamble bit, TxBD[P].
10	—	Reserved, should be cleared.
11	PEN	Parity enable 0 No parity 1 Parity is enabled and determined by the parity mode bits.
12–13, 14–15	RPM, TPM	Receiver/transmitter parity mode. Selects the type of parity check the receiver/transmitter performs; can be modified on-the-fly. Receive parity errors can be ignored but not disabled. 00 Odd parity. If a transmitter counts an even number of ones in the data word, it sets the parity bit so an odd number is sent. If a receiver receives an even number, a parity error is reported. 01 Low parity (space parity). A transmitter sends a zero in the parity bit position. If a receiver does not read a 0 in the parity bit, a parity error is reported. 10 Even parity. Like odd parity, the transmitter adjusts the parity bit, as necessary, to ensure that the receiver receives an even number of one bits; otherwise, a parity error is reported. 11 High parity (mark parity). The transmitter sends a one in the parity bit position. If the receiver does not read a 1 in the parity bit, a parity error is reported.

29.17 SCC UART Receive Buffer Descriptor (RxBD)

The CPM uses RxBDs to report on each buffer received. The CPM closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- A user-defined control character is received.
- An error occurs during message processing.
- A full receive buffer is detected.
- A MAX_IDL number of consecutive idle characters is received.
- An ENTER HUNT MODE or CLOSE RXBD command is issued.

- An address character is received in multidrop mode. The address character is written to the next buffer for a software comparison.

Figure 29-7 shows an example of how RxBDs are used in receiving.

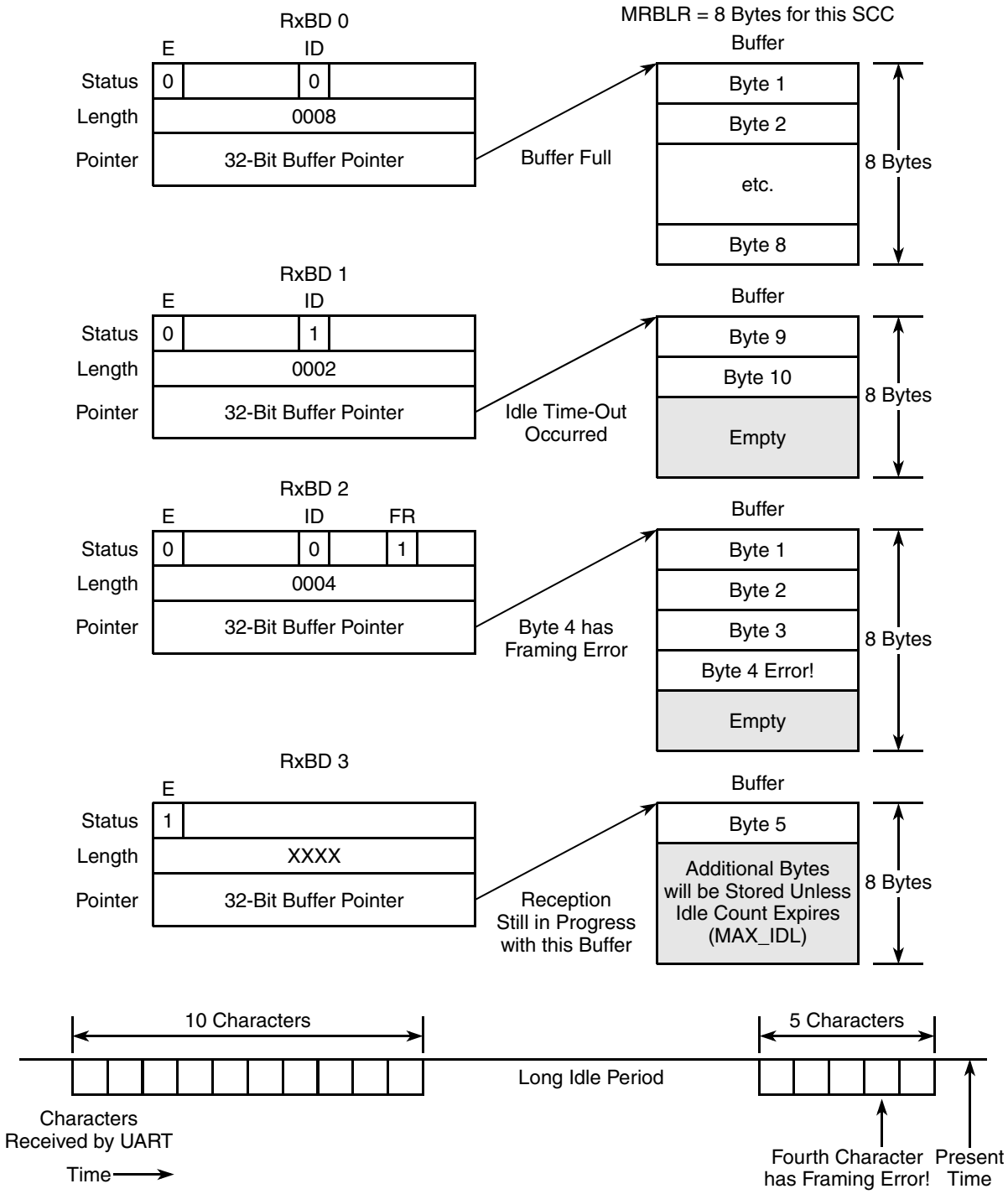


Figure 29-7. SCC UART Receiving Using RxBDs

Figure 29-8 shows the SCC UART RxBD.

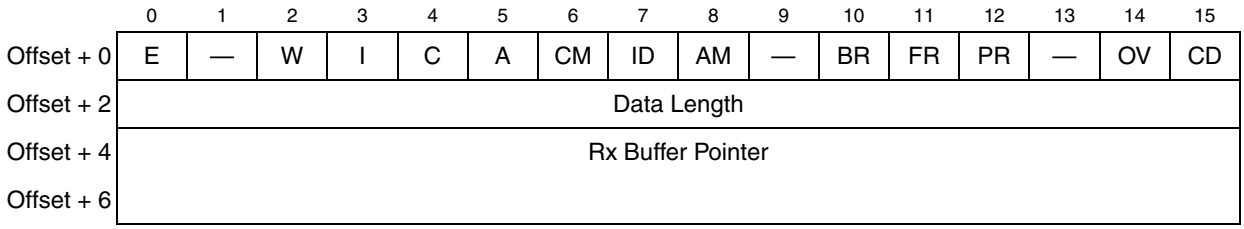


Figure 29-8. SCC UART Receive Buffer Descriptor (RxBD)

Table 29-10 describes RxBD status and control fields.

Table 29-10. SCC UART RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty 0 The buffer is full or reception was aborted due to an error. The core can read or write to any fields of this BD. The CPM does not reuse this BD while E = 0. 1 The buffer is not full. The CPM controls this BD and buffer. The core should not modify this BD.
1	—	Reserved, should be cleared.
2	W	Wrap (last buffer descriptor in the BD table) 0 Not the last descriptor in the table 1 Last descriptor in the table. After this buffer is used, the CPM receives incoming data using the BD pointed to by RBASE. The number of BDs in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is filled. 1 The CP sets SCCE[RX] when this buffer is completely filled by the CPM, indicating the need for the core to process the buffer. Setting SCCE[RX] causes an interrupt if not masked.
4	C	Control character 0 This buffer does not contain a control character. 1 The last byte in this buffer matches a user-defined control character.
5	A	Address 0 The buffer contains only data. 1 For manual multidrop mode, A indicates the first byte of this buffer is an address byte. Software should perform address comparison. In automatic multidrop mode, A indicates the buffer contains a message received immediately after an address matched UADDR1 or UADDR2. The address itself is not written to the buffer but is indicated by the AM bit.
6	CM	Continuous mode 0 Normal operation. The CPM clears E after this BD is closed. 1 The CPM does not clear E after this BD is closed, allowing the buffer to be overwritten when the CPM accesses this BD again. E is cleared if an error occurs during reception, regardless of CM.
7	ID	Buffer closed on reception of idles. The buffer is closed because a programmable number of consecutive idle sequences (MAX_IDL) was received.
8	AM	Address match. Significant only if the address bit is set and automatic multidrop mode is selected in PSMR[UM]. After an address match, AM identifies which user-defined address character was matched. 0 The address matched the value in UADDR2. 1 The address matched the value in UADDR1.

Table 29-10. SCC UART RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
9	—	Reserved, should be cleared.
10	BR	Break received. Set when a break sequence is received as data is being received into this buffer
11	FR	Framing error. Set when a character with a framing error (a character without a stop bit) is received and located in the last byte of this buffer. A new Rx buffer is used to receive subsequent data.
12	PR	Parity error. Set when a character with a parity error is received and located in the last byte of this buffer. A new Rx buffer is used to receive subsequent data.
13	—	Reserved, should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception
15	CD	Carrier detect lost. Set when the carrier detect signal is negated during reception

Section 28.3, "SCC Buffer Descriptors (BDs)," describes the data length and buffer pointer fields.

29.18 SCC UART Transmit Buffer Descriptor (TxBD)

The CPM uses BDs to confirm transmission and indicate error conditions so the core knows that buffers have been serviced. Figure 29-9 shows the SCC UART TxBD.

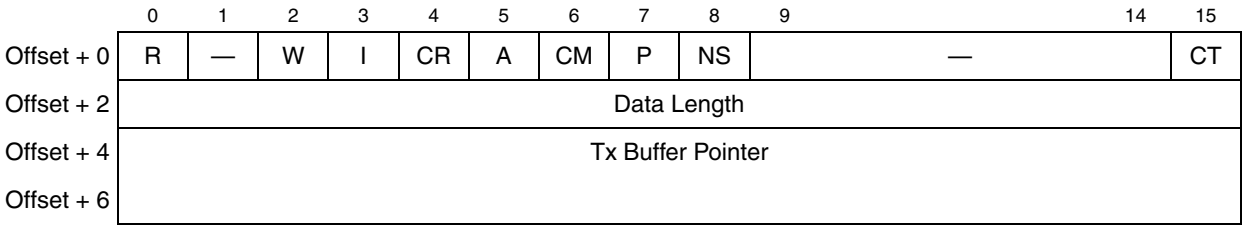


Figure 29-9. SCC UART Transmit Buffer Descriptor (TxBD)

Table 29-11 describes TxBD status and control fields.

Table 29-11. SCC UART TxBD Status and Control Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer is not ready. This BD and buffer can be modified. The CPM automatically clears R after the buffer is sent or an error occurs. 1 The user-prepared buffer is waiting to begin transmission or is being transmitted. Do not modify the BD once R is set.
1	—	Reserved, should be cleared.
2	W	Wrap (last buffer descriptor in TxBD table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined only by the W bit and space constraints of the dual-port RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is processed. 1 SCCE[TX] is set after this buffer is processed by the CPM, which can cause an interrupt.

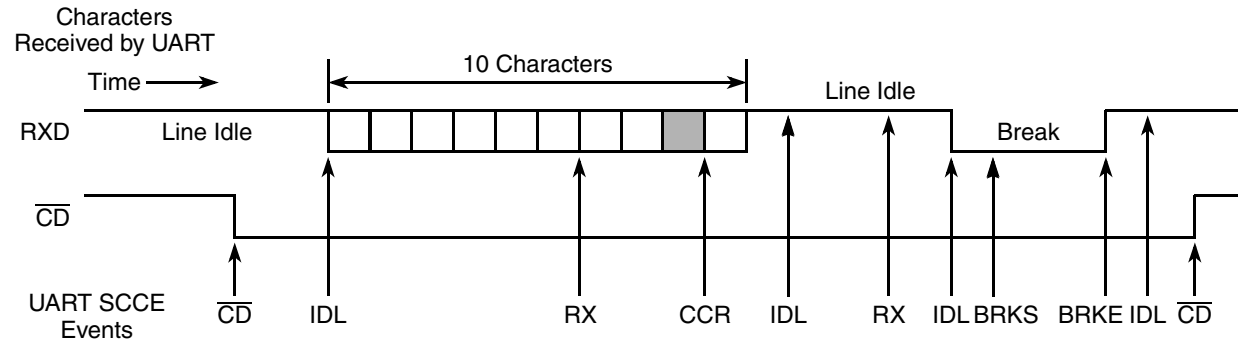
Table 29-11. SCC UART TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
4	CR	Clear-to-send report 0 The next buffer is sent with no delay (assuming it is ready), but if a $\overline{\text{CTS}}$ lost condition occurs, TxBD[CT] may not be set in the correct TxBD or may not be set at all. Asynchronous flow control, however, continues to function normally. 1 Normal $\overline{\text{CTS}}$ lost error reporting and three bits of idle are sent between consecutive buffers.
5	A	Address. Valid only in multidrop mode—automatic or manual 0 This buffer contains only data. 1 This buffer contains address characters. All data in this buffer is sent as address characters.
6	CM	Continuous mode 0 Normal operation. The CPM clears R after this BD is closed. 1 The CPM does not clear R after this BD is closed, allowing the buffer to be resent next time the CPM accesses this BD. However, R is cleared by transmission errors, regardless of CM.
7	P	Preamble 0 No preamble sequence is sent. 1 Before sending data, the controller sends an idle character consisting of all ones. If the data length of this BD is zero, only a preamble is sent.
8	NS	No stop bit or shaved stop bit sent. 0 Normal operation. Stop bits are sent with all characters in this buffer. 1 If PSMR[SYN] = 1, data in this buffer is sent without stop bits. If SYN = 0, the stop bit is shaved, depending on the DSR setting; see Section 29.14, “Fractional Stop Bits (Transmitter).”
9–14	—	Reserved, should be cleared.
15	CT	$\overline{\text{CTS}}$ lost. The CPM writes this status bit after sending the associated buffer. 0 $\overline{\text{CTS}}$ remained asserted during transmission. 1 $\overline{\text{CTS}}$ negated during transmission.

The data length and buffer pointer fields are described in Section 28.3, “SCC Buffer Descriptors (BDs).”

29.19 SCC UART Event Register (SCCE) and Mask Register (SCCM)

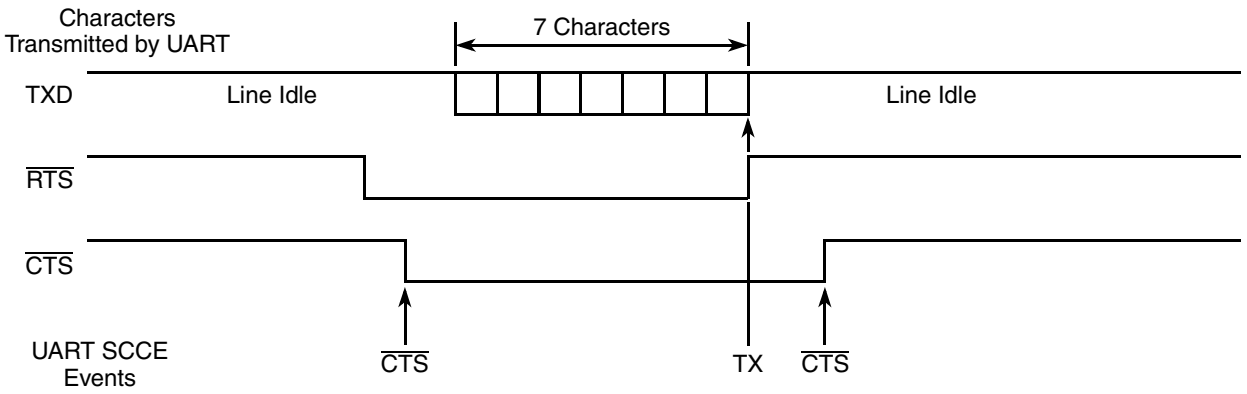
The SCC event register (SCCE) is used to report events recognized by the UART channel and to generate interrupts. When an event is recognized, the controller sets the corresponding SCCE bit. Interrupts can be masked in the UART mask register (SCCM), which has the same format as SCCE. Setting a mask bit enables the corresponding SCCE interrupt; clearing a bit masks it. Figure 29-10 shows example interrupts that can be generated by the SCC UART controller.



Notes:
1. The first RX event assumes Rx buffers are 6 bytes each.
2. The second IDL event occurs after an all-ones character is received.
3. The second RX event position is programmable based on the MAX_IDL value.
4. The BRKS event occurs after the first break character is received.
5. The $\overline{\text{CD}}$ event must be programmed in the port C parallel I/O, not in the SCC itself.

Legend:

A receive control character defined not to be stored in the Rx buffer.



Notes:
1. TX event assumes all seven characters were put into a single buffer and TxBD[CR] = 1.
2. The $\overline{\text{CTS}}$ event must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 29-10. SCC UART Interrupt Event Example

SCCE bits are cleared by writing ones; writing zeros has no effect. Unmasked bits must be cleared before the CPM clears an internal interrupt request. Figure 29-11 shows SCCE/SCCM for UART operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—						AB	IDL	GRA	BRKE	BRKS	—	CCR	BSY	TX	RX
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1A10 (SCCE1); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)															

Figure 29-11. SCC UART Event Register (SCCE) and Mask Register (SCCM)

Table 29-12 describes SCCE fields for UART mode.

Table 29-12. SCCE/SCCM Field Descriptions for UART Mode

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	AB	Autobaud. Set when an autobaud lock is detected. The core should rewrite the baud rate generator with the precise divider value. See Chapter 25, "Baud-Rate Generators (BRGs)."
7	IDL	Idle sequence status changed. Set when the channel detects a change in the serial line. The line's real-time status can be read in SCCS[ID]. Idle is entered when a character of all ones is received; it is exited when a zero is received.
8	GRA	Graceful stop complete. Set as soon as the transmitter finishes any buffer in progress after a GRACEFUL STOP TRANSMIT command is issued. It is set immediately if no buffer is in progress.
9	BRKE	Break end. Set when an idle bit is received after a break sequence.
10	BRKS	Break start. Set when the first character of a break sequence is received. Multiple BRKS events are not received if a long break sequence is received.
11	—	Reserved, should be cleared.
12	CCR	Control character received and rejected. Set when a control character is recognized and stored in the receive control character register RCCR.
13	BSY	Busy. Set when a character is received and discarded due to a lack of buffers. In multidrop mode, the receiver automatically enters hunt mode; otherwise, reception continues when a buffer is available. The latest point that an RxBD can be changed to empty and guarantee avoiding the busy condition is the middle of the stop bit of the first character to be stored in that buffer.
14	TX	Tx event. Set when a buffer is sent. If TxBD[CR] = 1, TX is set no sooner than when the last stop bit of the last character in the buffer begins transmission. If TxBD[CR] = 0, TX is set after the last character is written to the Tx FIFO. TX also represents a $\overline{\text{CTS}}$ lost error; check TxBD[CT].
15	RX	Rx event. Set when a buffer is received, which is no sooner than the middle of the first stop bit of the character that caused the buffer to close. Also represents a general receiver error (overrun, $\overline{\text{CD}}$ lost, parity, idle sequence, and framing errors); the RxBD status and control fields indicate the specific error.

29.20 SCC UART Status Register (SCCS)

The SCC UART status register (SCCS), shown in Figure 29-12, monitors the real-time status of RXD.

	0 — 6	7
Field	—	ID
Reset	0000_0000_0000_0000	
R/W	R	
Offset	0x9_1A17 (SCCS1); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)	

Figure 29-12. SCC Status Register for UART Mode (SCCS)

Table 29-13 describes UART SCCS fields.

Table 29-13. UART SCCS Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	ID	Idle status. Set when RXD has been a logic one for at least a full character time. 0 The line is not idle. 1 The line is idle.

29.21 S-Records Loader Application

This section describes a downloading application that uses an SCC UART controller. The application performs S-record downloads and uploads between a host computer and an intelligent peripheral through a serial asynchronous line. S-records are strings of ASCII characters that begin with ‘S’ and end in an end-of-line character. This characteristic is used to impose a message structure on the communication between the devices. For flow control, each device can transmit XON and XOFF characters, which are not part of the program being uploaded or downloaded.

Table 29-14. UART Control Characters for S-Records Example

Character	Description
Line Feed	Both the E and R bits should be cleared. When an end-of-line character is received, the current buffer is closed and made available to the core for processing. This buffer contains an entire S record that the processor can now check and copy to memory or disk as required.
XOFF	E should be cleared; R should be set. Whenever the core receives a control-character-received (CCR) interrupt and the RCCR contains XOFF, the software should immediately stop transmitting by setting PSMR[FRZ]. This keeps the other station from losing data when it runs out of Rx buffers.
XON	XON should be received after XOFF. E should be cleared and R should be set. PSMR[FRZ] on the transmitter should now be cleared. The CPM automatically resumes transmission of the serial line at the point at which it was previously stopped. Like XOFF, the XON character is not stored in the receive buffer.

To receive S-records, the core must wait for an RX interrupt, indicating that a complete S-record buffer was received. Transmission requires assembling S-records into buffers and linking them to the TxBD table; transmission can be paused when an XOFF character is received. This scheme minimizes the number of interrupts the core receives (one per S-record) and relieves it from continually scanning for control characters.

Chapter 30
SCC HDLC Mode

High-level data link control (HDLC) is one of the most common protocols in the data link layer, layer 2 of the OSI model. Many other common layer-2 protocols, such as SDLC, SS#7, AppleTalk, LAPB, and LAPD, are based on HDLC and its framing structure in particular. Figure 30-1 shows the HDLC framing structure.

HDLC uses a zero insertion/deletion process (bit-stuffing) to ensure that a data bit pattern matching the delimiter flag does not occur in a field between flags. The HDLC frame is synchronous and relies on the physical layer for clocking and synchronization of the transmitter/receiver.

An address field is needed to carry the frame's destination address because the layer 2 frame can be sent over point-to-point links, broadcast networks, packet-switched or circuit-switched systems. An address field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. SDLC and LAPB use an 8-bit address. SS#7 has no address field because it is always used in point-to-point signaling links. LAPD divides its 16-bit address into different fields to specify various access points within one device. LAPD also defines a broadcast address. Some HDLC-type protocols permit addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow control number and defines the frame type (control or data). The exact use and structure of this field depends on the protocol using the frame. The length of the data in the data field depends on the frame protocol. Layer 3 frames are carried in this data field. Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16 bits long but can be as long as 32 bits. In HDLC, the lsb of each octet is sent first; the msb of the CRC is sent first.

HDLC mode is selected for an SCC by writing GSMR_L[MODE] = 0b0000. In a nonmultiplexed modem interface, SCC outputs connect directly to external pins. Modem signals can be supported through port C. The Rx and Tx clocks can be supplied from either the bank of baud rate generators, by the DPLL, or externally. An SCC can also be connected through the TDM channels of the serial interface (SI). In HDLC mode, an SCC becomes an HDLC controller, and consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to other SCCs.

30.1 SCC HDLC Features

The main features of an SCC in HDLC mode are follows:

- Flexible buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (Rx and Tx)
- Received-frames threshold to reduce interrupt overhead
- Can be used with the SCC DPLL

- Four address comparison registers with mask
- Maintenance of five 16-bit error counters
- Flag/abort/idle generation and detection
- Zero insertion/deletion
- 16- or 32-bit CRC-CCITT generation and checking
- Detection of nonoctet aligned frames
- Detection of frames that are too long
- Programmable flags (0–15) between successive frames
- Automatic retransmission in case of collision

30.2 SCC HDLC Channel Frame Transmission

The HDLC transmitter is designed to work with little or no core intervention. Once enabled by the core, a transmitter starts sending flags or idles as programmed in the HDLC mode register (PSMR). The HDLC polls the first BD in the TxBD table. When there is a frame to transmit, the SCC fetches the data (address, control, and information) from the first buffer and starts sending the frame after inserting the minimum number of flags specified between frames. When the end of the current buffer is reached and TxBD[L] (last buffer in frame) is set, the SCC appends the CRC and closing flag. In HDLC mode, the lsb of each octet and the msb of the CRC are sent first. Figure 30-1shows a typical HDLC frame.

Opening Flag	Address	Control	Information (Optional)	CRC	Closing Flag
8 bits	16 bits	8 bits	8*n* bits	16 bits	8 bits

Figure 30-1. HDLC Framing Structure

After a closing flag is sent, the SCC updates the frame status bits of the BD and clears TxBD[R] (buffer ready). At the end of the current buffer, if TxBD[L] is not set (multiple buffers per frame), only TxBD[R] is cleared. Before the SCC proceeds to the next TxBD in the table, an interrupt can be issued if TxBD[I] is set. This interrupt programmability allows the core to intervene after each buffer, after a specific buffer, or after each frame.

The STOP TRANSMIT command can be used to expedite critical data ahead of previously linked buffers or to support efficient error handling. When the SCC receives a STOP TRANSMIT command, it sends idles or flags instead of the current frame until it receives a RESTART TRANSMIT command. The GRACEFUL STOP TRANSMIT command can be used to insert a high-priority frame without aborting the current one—a graceful-stop-complete event is generated in SCCE[GRA] when the current frame is finished. See Section 30.6, “SCC HDLC Commands.”

30.3 SCC HDLC Channel Frame Reception

The HDLC receiver is designed to work with little or no core intervention to perform address recognition, CRC checking, and maximum frame length checking. Received frames can be used to implement any HDLC-based protocol.

Once enabled by the core, the receiver waits for an opening flag character. When it detects the first byte of the frame, the SCC compares the frame address with four user-programmable, 16-bit address registers and an address mask. The SCC compares the received address field with the user-defined values after masking with the address mask. To detect broadcast (all ones) address frames, one address register must be written with all ones.

If an address match is detected, the SCC fetches the next BD and SCC starts transferring the incoming frame to the buffer if it is empty. When the buffer is full, the SCC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. If the incoming frame is larger than the current buffer, the SCC continues receiving using the next BD in the table.

During reception, the SCC checks for frames that are too long (using MFLR). When the frame ends, the CRC field is checked against the recalculated value and written to the buffer. RxBD[Data Length] of the last BD in the HDLC frame contains the entire frame length. This also enables software to identify the frames in which the maximum frame length violations occur. The SCC sets RxBD[L] (last buffer in frame), writes the frame status bits, and clears RxBD[E]. It then generates a maskable event (SCCE[RXF]) to indicate a frame was received. The SCC then waits for a new frame. Back-to-back frames can be received with only one shared flag between frames.

The received frames threshold parameter (RFTHR) can be used to postpone interrupts until a specified number of frames is received. This function can be combined with a timer to implement a timeout if fewer than the specified number of threshold frames is received.

Note that SCCs in HDLC mode, or any other synchronous mode, must receive a minimum of eight clocks after the last bit arrives to account for Rx FIFO delay.

30.4 SCC HDLC Parameter RAM

For HDLC mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 30-1.

Table 30-1. HDLC-Specific SCC Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**C_MASK**	Word	CRC mask. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8. For 32-bit CRC-CCITT, initialize with 0xDEBB_20E3.
0x38	**C_PRES**	Word	CRC preset. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF. For 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF.
0x3C	**DISFC**	Hword	Modulo 2^{16} counters maintained by the CP. Initialize them while the channel is disabled. DISFC (Discarded frame counter) Counts error-free frames discarded due to lack of free buffers. CRCEC (CRC error counter) Includes frames not addressed to the user or frames received in the BSY condition, but does not include overrun errors. ABTSC (Abort sequence counter) NMARC (Nonmatching address received counter) Includes error-free frames only. RETRC (Frame retransmission counter) Counts number of frames resent due to collision.
0x3E	**CRCEC**	Hword	
0x40	**ABTSC**	Hword	
0x42	**NMARC**	Hword	
0x44	**RETRC**	Hword	

Table 30-1. HDLC-Specific SCC Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
0x46	**MFLR**	Hword	Max frame length register. The HDLC compares the incoming HDLC frame's length with the user-defined limit in MFLR. If the limit is exceeded, the rest of the frame is discarded and RxBD[LG] is set in the last BD of that frame. At the end of the frame the SCC reports frame status and frame length in the last RxBD. The MFLR is defined as all in-frame bytes between the opening and closing flags.
0x48	MAX_CNT	Hword	Maximum length counter. A temporary down-counter used to track frame length.
0x4A	**RFTHR**	Hword	Received frames threshold. Used to reduce potential interrupt overhead when each in a series of short HDLC frames causes an SCCE[RXF] event. Setting RFTHR determines the frequency of RXF interrupts, which occur only when the RFTHR limit is reached. Provide enough empty RxBDs for the number of frames specified in RFTHR.
0x4C	RFCNT	Hword	Received frames count. RFCNT is a down-counter used to implement RFTHR.
0x4E	**HMASK**	Hword	Mask register (HMASK) and four address registers (HADDR*n*) for address recognition. The SCC reads the frame address from the HDLC receiver, compares it with the HADDRs, and masks the result with HMASK. Setting an HMASK bit enables the corresponding comparison bit, clearing a bit masks it. When a match occurs, the frame address and data are written to the buffers. When no match occurs and a frame is error-free, the nonmatching address received counter (NMARC) is incremented. The eight low-order bits of HADDR*n* should contain the first address byte after the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDR*n* should contain 0xAA68 and HMASK should contain 0xFFFF. For 8-bit addresses, clear the eight high-order HMASK bits. See Figure 30-2.
0x50	**HADDR1**	Hword	
0x52	**HADDR2**	Hword	
0x54	**HADDR3**	Hword	
0x56	**HADDR4**	Hword	
0x58	TMP	Hword	Temporary storage
0x5A	TMP_MB	Hword	Temporary storage

[1] From SCC base. See Section 28.4.1, "SCC Base Addresses."

Figure 30-2 shows 16- and 8-bit address recognition.

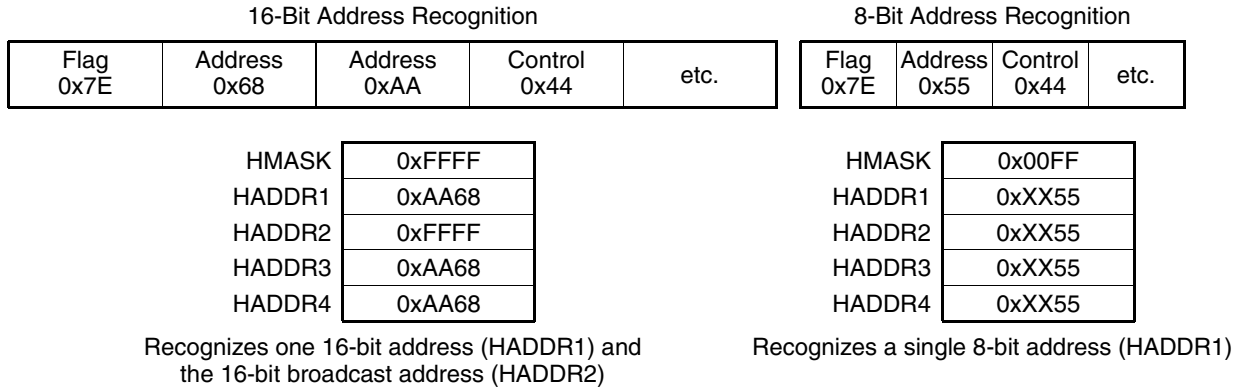


Figure 30-2. HDLC Address Recognition

30.5 Programming the SCC in HDLC Mode

HDLC mode is selected for an SCC by writing GSMR_L[MODE] = 0b0000. The HDLC controller uses the same buffer and BD data structure as other modes and supports multi buffer operation and address

comparisons. Receive errors are reported through the RxBD; transmit errors are reported through the TxBD.

30.6 SCC HDLC Commands

The transmit and receive commands are issued to the CP command register (CPCR). Transmit commands are described in Table 30-2.

Table 30-2. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GSMR, the transmitter starts polling the first BD in the TxBD table every 64 Tx clocks, or immediately if TODR[TOD] = 1, and begins sending data if TxBD[R] is set. If the SCC receives the STOP TRANSMIT command while not transmitting, the transmitter stops polling the BDs. If the SCC receives the command during transmission, transmission is aborted after a maximum of 64 additional bits, the Tx FIFO is flushed, and the current BD pointer TBPTR is not advanced (no new BD is accessed). The transmitter then sends an abort sequence (0x7F) and stops polling the BDs. When not transmitting, the channel sends flags or idles as programmed in the GSMR. Note that if PSMR[MFF] = 1, multiple small frames could be flushed from the Tx FIFO; a GRACEFUL STOP TRANSMIT command prevents this.
GRACEFUL STOP TRANSMIT	Stops transmission smoothly. Unlike a STOP TRANSMIT command, it stops transmission after the current frame is finished or immediately if no frame is being sent. SCCE[GRA] is set when transmission stops. HDLC Tx parameters and TxBDs can then be updated. TBPTR points to the next TxBD. Transmission begins once TxBD[R] of the next BD is set and a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Enables frames to be sent on the transmit channel. The HDLC controller expects this command after a STOP TRANSMIT is issued and the channel in its GSMR is disabled, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error. The transmitter resumes from the current BD.
INIT TX PARAMETERS	Resets the Tx parameters in the parameter RAM. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

Receive commands are described in Table 30-3.

Table 30-3. Receive Commands

Command	Description
ENTER HUNT MODE	After a hardware or software reset, once an SCC is enabled in the GSMR, the receiver is automatically enabled and uses the first BD in the RxBD table. While the SCC is looking for the beginning of a frame, that SCC is in hunt mode. The ENTER HUNT MODE command is used to force the HDLC receiver to stop receiving the current frame and enter hunt mode, in which the HDLC continually scans the input data stream for a flag sequence. After receiving the command, the buffer is closed and the CRC is reset. Further frame reception uses the next BD.
CLOSE RXBD	Should not be used in the HDLC protocol.
INIT RX PARAMETERS	Resets the Rx parameters in the parameter RAM.; issue only when the receiver is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

30.7 Handling Errors in the SCC HDLC Controller

The SCC HDLC controller reports frame reception and transmission errors using BDs, error counters, and the SCCE. Transmission errors are described in Table 30-4.

Table 30-4. Transmit Errors

Error	Description
Transmitter Underrun	The channel stops transmitting, closes the buffer, sets TxBD[UN], and generates a TXE interrupt if not masked. Transmission resumes when a RESTART TRANSMIT command is issued. The SCC send and receive FIFOs are 32 bytes each.
$\overline{\text{CTS}}$ Lost during Frame Transmission	The channel stops transmitting, closes the buffer, sets TxBD[CT], and generates the TXE interrupt if not masked. Transmission resumes after a RESTART TRANSMIT command. If this error occurs on the first or second buffer of the frame and PSMR[RTE] = 1, the channel resends the frame when $\overline{\text{CTS}}$ is reasserted and no error is reported. If collisions are possible, to ensure proper retransmission of multi-buffer frames, the first two buffers of each frame should in total contain more than 36 bytes for SCC or 20 bytes for SCC. The channel also increments the retransmission counter RETRC in the parameter RAM.

Reception errors are described in Table 30-5.

Table 30-5. Receive Errors

Error	Description
Overrun	Each SCC maintains an internal FIFO for receiving data. The CP begins programming the SDMA channel (if the buffer is in external memory) and updating the CRC when a full or partial FIFO's worth of data (according to GSMR_H[RFW]) is received in the Rx FIFO. When an Rx FIFO overrun occurs, the previous byte is overwritten by the next byte. The previous data byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates an RXF interrupt if not masked. The receiver then enters hunt mode. Even if an overrun occurs during a frame whose address is not recognized, an RxBD with data length two is opened to report the overrun and the interrupt is generated.
$\overline{\text{CD}}$ Lost during Frame Reception	Highest priority error. The channel stops frame reception, closes the buffer, sets RxBD[CD], and generates the RXF interrupt if not masked. The rest of the frame is lost and other errors are not checked in that frame. At this point, the receiver enters hunt mode.
Abort Sequence	Occurs when seven or more consecutive ones are received. When this occurs while receiving a frame, the channel closes the buffer, sets RxBD[AB] and generates a maskable RXF interrupt. The channel also increments the abort sequence counter ABTSC. The CRC and nonoctet error status conditions are not checked on aborted frames. The receiver then enters hunt mode.

Table 30-5. Receive Errors (continued)

Error	Description
Nonoctet Aligned Frame	The channel writes the received data to the buffer, closes the buffer, sets RxBD[NO], and generates a maskable RXF interrupt. CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. An immediate back-to-back frame is still received. The nonoctet data may be derived from the last word in the buffer as follows: msb … lsb … | 1 | 0 … 0 Valid Data | Nonvalid Data Note that if buffer swapping is used (RFCR[BO] = 0b0x), the figure above refers to the last byte, rather than the last word, of the buffer. The lsb of each octet is sent first while the msb of the CRC is sent first.
CRC	The channel writes the received CRC to the buffer, closes the buffer, sets RxBD[CR], generates a maskable RXF interrupt, and increments the CRC error counter CRCEC. After receiving a frame with a CRC error, the receiver enters hunt mode. An immediate back-to-back frame is still received. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

30.8 HDLC Mode Register (PSMR)

The protocol-specific mode register (PSMR), shown in Figure 30-3, functions as the HDLC mode register.

	0–3	4–5	6	7	8	9	10	11	12	13–15
Field	NOF	CRC	RTE	—	FSE	DRT	BUS	BRM	MFF	—
Reset	0									
R/W	R/W									
Address	0x9_1A08 (PSMR1); 0x9_1A48 (PSMR3); 0x9_1A68 (PSMR4)									

Figure 30-3. HDLC Mode Register (PSMR)

Table 30-6 describes PSMR HDLC fields.

Table 30-6. PSMR HDLC Field Descriptions

Bits	Name	Description
0–3	NOF	Number of flags. Minimum number of flags between or before frames. If NOF = 0b0000, no flags are inserted between frames and the closing flag of one frame is followed by the opening flag of the next frame in the case of back-to-back frames. NOF can be modified on-the-fly.
4–5	CRC	CRC selection. 00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1 x1 Reserved 10 32-bit CCITT-CRC (HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1
6	RTE	Retransmit enable 0 No retransmission 1 Automatic frame retransmission is enabled. Particularly useful in the HDLC bus protocol and ISDN applications where multiple HDLC controllers can collide. Note that retransmission occurs only if a lost $\overline{\text{CTS}}$ occurs on the first or second buffer of the frame.

Table 30-6. PSMR HDLC Field Descriptions (continued)

Bits	Name	Description
7	—	Reserved, should be cleared
8	FSE	Flag sharing enable. FSE can be set only if GSMR_H[RTSM] is already set. Can be modified on-the-fly 0 Normal operation 1 If NOF[0–3] = 0b0000, a single shared flag is sent between back-to-back frames. Other values of NOF[0–3] are decremented by 1. Useful in signaling system #7 applications
9	DRT	Disable receiver while transmitting 0 Normal operation 1 As the SCC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$. This helps if the HDLC channel is on a multidrop line and the SCC does not need to receive its own transmission.
10	BUS	HDLC bus mode 0 Normal HDLC operation 1 HDLC bus operation is selected. See Section 30.13, “HDLC Bus Mode with Collision Detection.”
11	BRM	HDLC bus $\overline{\text{RTS}}$ mode. Valid only if BUS = 1. Otherwise, it is ignored. 0 Normal $\overline{\text{RTS}}$ operation during HDLC bus mode. $\overline{\text{RTS}}$ is asserted on the first bit of the Tx frame and negated after the first collision bit is received. 1 Special $\overline{\text{RTS}}$ operation during HDLC bus mode. $\overline{\text{RTS}}$ is delayed by one bit with respect to the normal case, which helps when the HDLC bus protocol is being run locally and sent over a long-distance line at the same time. The one-bit delay allows $\overline{\text{RTS}}$ to be used to enable the transmission line buffers so that the electrical effects of collisions are not sent over the transmission line.
12	MFF	Multiple frames in Tx FIFO. The receiver is not affected. 0 Normal operation. The Tx FIFO must never contain more than one HDLC frame. The $\overline{\text{CTS}}$ lost status is reported accurately on a per-frame basis. 1 The Tx FIFO can hold multiple frames, but lost $\overline{\text{CTS}}$ may not be reported on the buffer/frame it occurred on. This can improve performance of HDLC transmissions of small back-to-back frames or when the number of flags between frames should be limited.
13–15	—	Reserved, should be cleared.

30.9 SCC HDLC Receive Buffer Descriptor (RxBD)

The CP uses the RxBD, shown in Figure 30-4, to report on data received for each buffer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	F	**CM**	—	DE	—	LG	NO	AB	CR	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Buffer Pointer**															
Offset + 6																

Figure 30-4. SCC HDLC Receive Buffer Descriptor (RxBD)

Table 30-7 describes HDLC RxBD status and control fields.

Table 30-7. SCC HDLC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The buffer is full or reception stopped because of an error. The core can read or write to any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer is not full. The CP controls the BD and buffer. The core should not update the BD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in the RxBD table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE. The number of BDs in this table are programmable and determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 SCCE[RXB] is not set after this buffer is used; SCCE[RXF] is unaffected. 1 SCCE[RXB] or SCCE[RXF] is set when the SCC uses this buffer.
4	L	Last buffer in frame 0 Not the last buffer in frame 1 Last buffer in frame. Indicates reception of a closing flag or an error, in which case one or more of the CD, OV, AB, and LG bits are set. The SCC writes the number of frame octets to the data length field.
5	F	First in frame 0 Not the first buffer in a frame 1 First buffer in a frame
6	**CM**	Continuous mode. Note that RxBD[E] is cleared if an error occurs during reception, regardless of CM. 0 Normal operation 1 RxBD[E] is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten next time the CP accesses it.
7	—	Reserved, should be cleared.
8	DE	DPLL error. Set when a DPLL error occurs while this buffer is being received. DE is also set due to a missing transition when using decoding modes in which a transition is required for every bit. Note that when a DPLL error occurs, the frame closes and error checking halts.
9	—	Reserved, should be cleared.
10	LG	Rx frame length violation. Set when a frame larger than the maximum defined for this channel is recognized. Only the maximum-allowed number of bytes (MFLR) is written to the buffer. This event is not reported until the buffer is closed, SCCE[RXF] is set, and the closing flag is received. The total number of bytes received between flags is still written to the data length field.
11	NO	Rx nonoctet aligned frame. Set when a received frame contains a number of bits not divisible by eight.
12	AB	Rx abort sequence. Set when at least seven consecutive ones are received during frame reception.
13	CR	Rx CRC error. Set when a frame contains a CRC error. CRC bytes received are always written to the Rx buffer.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost (NMSI mode only). Set when $\overline{CD}$ is negated during frame reception.

Data length and buffer pointer fields are described in Section 28.3, “SCC Buffer Descriptors (BDs).” Because HDLC is a frame-based protocol, RxBD[Data Length] of the last buffer of a frame contains the total number of frame bytes, including the 2 or 4 bytes for CRC. Figure 30-5 shows an example of how RxBDs are used in receiving.

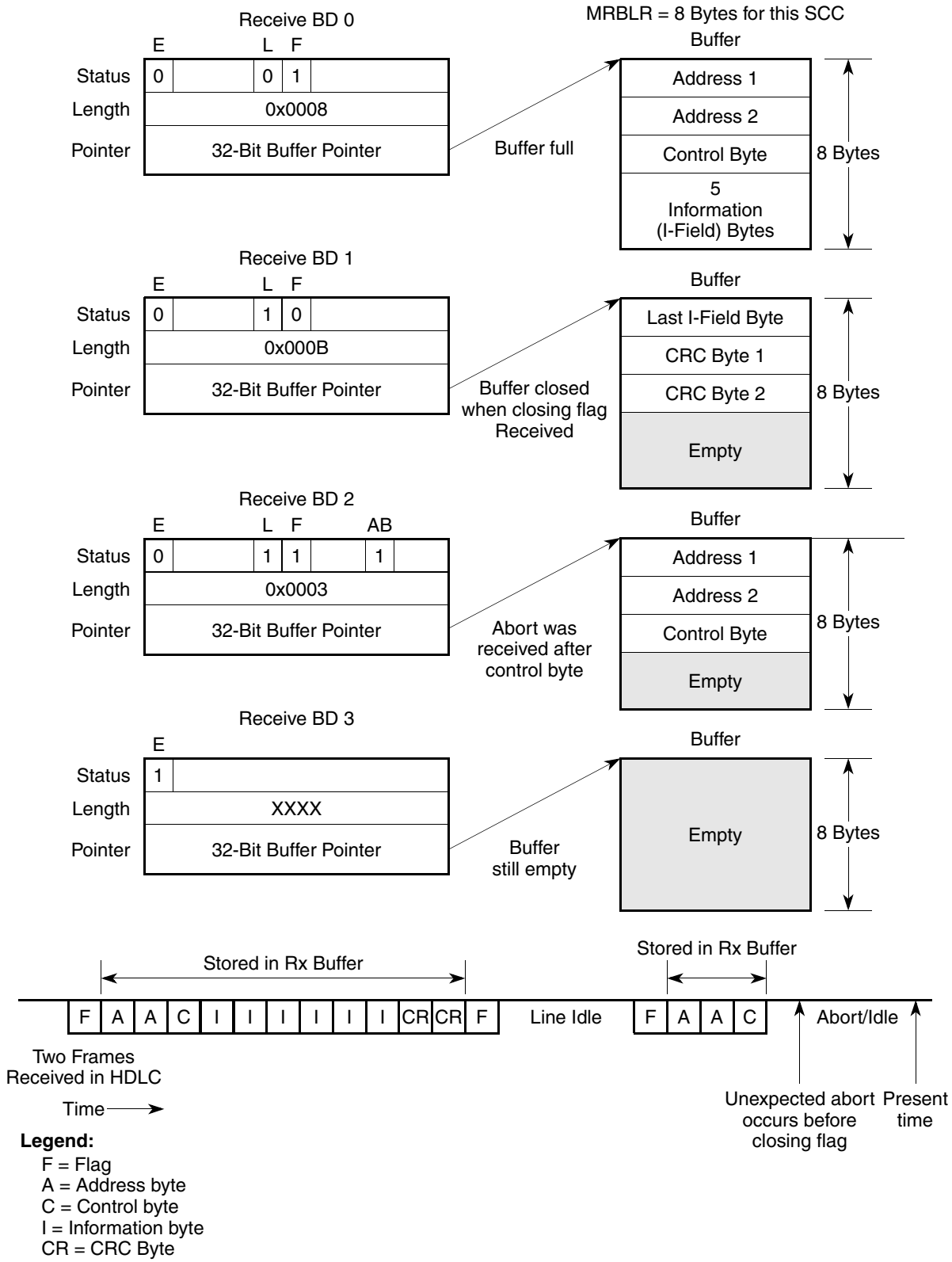


Figure 30-5. SCC HDLC Receiving Using RxBDs

30.10 SCC HDLC Transmit Buffer Descriptor (TxBD)

The CP uses the TxBD, shown in Figure 30-6, to confirm transmissions and indicate error conditions.

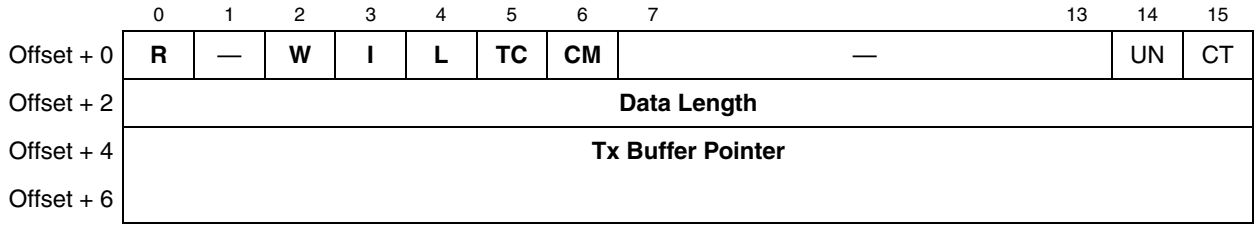


Figure 30-6. SCC HDLC Transmit Buffer Descriptor (TxBD)

Table 30-8 describes HDLC TxBD status and control fields.

Table 30-8. SCC HDLC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready 0 The buffer is not ready for transmission. Both the buffer and the BD can be updated. The CP clears R after the buffer is sent or an error is encountered. 1 The buffer has not been sent or is being sent and the BD cannot be updated.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in TxBD table) 0 Not the last BD in the table 1 Last BD in the BD table. After this buffer is used, the CP sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined by TxBD[W] and the space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is processed. 1 SCCE[TXB] or SCCE[TXE] is set when this buffer is processed, causing interrupts if not masked.
4	**L**	Last 0 Not the last buffer in the frame 1 Last buffer in the frame
5	**TC**	Tx CRC. Valid only when TxBD[L] = 1. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear TxBD[R] after this BD is closed allowing the buffer to be resent the next time the CP accesses this BD. However, TxBD[R] is cleared if an error occurs during transmission, regardless of CM.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. Set after the SCC sends a buffer and a transmitter underrun occurred
15	CT	$\overline{\text{CTS}}$ lost. Indicates when $\overline{\text{CTS}}$ in NMSI mode or layer 1 grant is lost in GCI or IDL mode during frame transmission. If data from more than one buffer is currently in the FIFO when this error occurs, the HDLC writes CT in the current BD after sending the buffer.

The data length and buffer pointer fields are described in Section 28.3, “SCC Buffer Descriptors (BDs).”

30.11 HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)

The SCC event register (SCCE) is used as the HDLC event register to report events recognized by the HDLC channel and to generate interrupts. When an event is recognized, the SCC sets the corresponding SCCE bit. Interrupts generated through SCCE can be masked in the SCC mask register (SCCM) which has the same bit format as the SCCE. Setting an SCCM bit enables the corresponding interrupt; clearing a bit masks it. SCCE bits are cleared by writing ones; writing zeros has no effect. All unmasked bits must be cleared before the CP clears the internal interrupt request. Figure 30-7 shows SCCE/SCCM for HDLC operation.

	0–4	5	6	7	8	9–10	11	12	13	14	15
Field	—	DCC	FLG	IDL	GRA	—	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000										
R/W	R/W										
Offset	0x9_1A10 (SCCE1); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)										

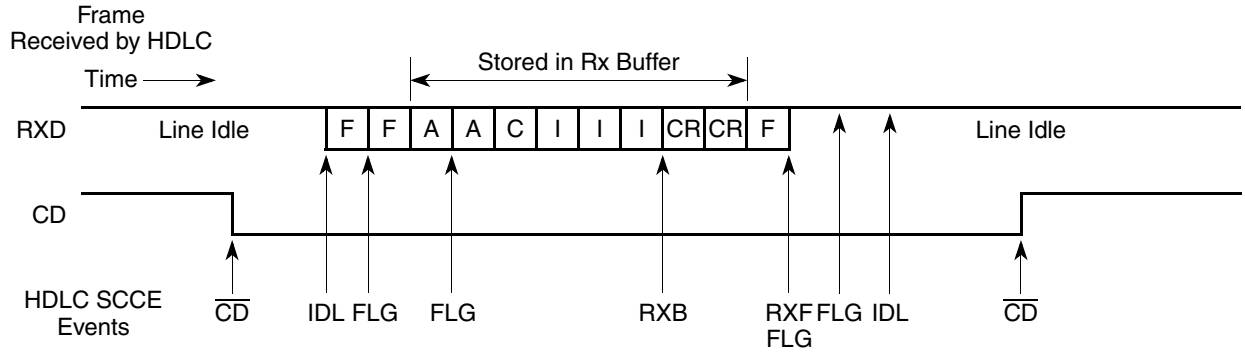
Figure 30-7. HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)

Table 30-9 describes SCCE/SCCM fields.

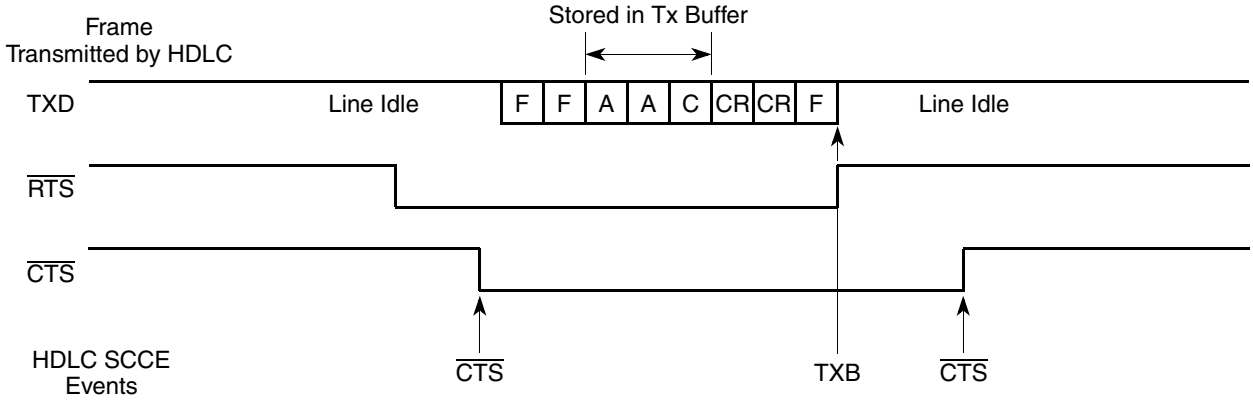
Table 30-9. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	DCC	DPLL carrier sense changed. Set when the carrier sense status generated by the DPLL changes. Real-time status can be read in SCCS[CS]. This is not the $\overline{CD}$ status reported in port C. Valid only when the DPLL is used.
6	FLG	Flag status. Set when the SCC stops or starts receiving HDLC flags. Real-time status can be read in SCCS[FG].
7	IDL	Idle sequence status changed. Set when HDLC line status changes. Real-time status of the line can be read in SCCS[ID].
8	GRA	Graceful stop complete. A GRACEFUL STOP TRANSMIT command completed execution. Set as soon as the transmitter has sent a frame in progress when the command was issued. Set immediately if no frame was in progress when the command was issued.
9–10	—	Reserved, should be cleared.
11	TXE	Tx error. Indicates an error ($\overline{CTS}$ lost or underrun) has occurred on the transmitter channel.
12	RXF	Rx frame. Set when the number of receive frames specified in RFTHR are received on the HDLC channel. It is set no sooner than two clocks after the last bit of the closing flag is received. This event is not maskable through the RxBD[I] bit.
13	BSY	Busy condition. Indicates a frame arrived but was discarded due to a lack of buffers.
14	TXB	Transmit buffer. Enabled by setting TxBD[I]. TXB is set when a buffer is sent on the HDLC channel. For the last buffer in the frame, TXB is not set before the last bit of the closing flag begins its transmission; otherwise, it is set after the last byte of the buffer is written to the Tx FIFO.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when the HDLC channel receives a buffer that is not the last in a frame.

Figure 30-8 shows interrupts that can be generated using the HDLC protocol.



Notes:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.
3. The FLG interrupts show the beginning and end of flag reception.
4. The FLG interrupt at the end of the frame may precede the RXF interrupt due to receive FIFO latency.
5. The CD event must be programmed in the port C parallel I/O, not in the SCC itself.
6. F = flag, A = address byte, C = control byte, I = information byte, and CR = CRC byte.



Notes:
1. TXB event shown assumes all three bytes were put into a single buffer.
2. Example shows one additional opening flag. This is programmable.
3. The CTS event must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 30-8. SCC HDLC Interrupt Event Example

30.12 SCC HDLC Status Register (SCCS)

The SCC status register (SCCS), shown in Figure 30-9, permits monitoring of real-time status conditions on RXD. The real-time status of CTS and CD are part of the port C parallel I/O.

	0 — 4	5	6	7
Field	—	FG	CS	ID
Reset	0000_0000			
R/W	R			
Offset	0x9_1A17 (SCCS1); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)			

Figure 30-9. SCC HDLC Status Register (SCCS)

Table 30-10 describes HDLC SCCS fields.

Table 30-10. HDLC SCCS Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	FG	Flags. The line is checked after the data has been decoded by the DPLL. 0 HDLC flags are not being received. The most recently received 8 bits are examined every bit time to see if a flag is present. 1 HDLC flags are being received. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once it is set, it remains set at least 8 bit times and the next eight received bits are examined. If another flag occurs, FG stays set for at least another eight bits. If not, it is cleared and the search begins again.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	ID	Idle status 0 The line is busy. 1 Set when RXD is a logic 1 (idle) for 15 or more consecutive bit times. It is cleared after a single logic 0 is received.

30.13 HDLC Bus Mode with Collision Detection

The HDLC controller includes an option for hardware collision detection and retransmission on an open-drain connected HDLC bus, referred to as HDLC bus mode. Most HDLC-based controllers provide only point-to-point communications; however, the HDLC bus enhancement allows implementation of an HDLC-based LAN and other point-to-multipoint configurations. The HDLC bus is based on techniques used in the CCITT ISDN I.430 and ANSI T1.605 standards for D-channel point-to-multipoint operation over the S/T interface. However, the HDLC bus does not fully comply with I.430 or T1.605 and cannot replace devices that implement these protocols. Instead, it is more suited to non-ISDN LAN and point-to-multipoint configurations.

Review the basic features of the I.430 and T1.605 before learning about the HDLC bus. The I.430 and T1.605 define a way to connect eight terminals over the D-channel of the S/T ISDN bus. The layer 2 protocol is a variant of HDLC, called LAPD. However, at layer 1, a method is provided to allow the eight terminals to send frames to the switch through the physical S/T bus.

To determine whether a channel is clear, the S/T interface device looks at an echo bit on the line designed to echo the last bit sent on the D channel. Depending on the class of terminal and the context, an S/T interface device waits for 7–10 ones on the echo bit before letting the LAPD frame begin transmission, after which the S/T interface monitors transmitted data. As long as the echo bit matches the sent data, transmission continues. If the echo bit is ever 0 when the transmit bit is 1, a collision occurs between terminals; the station(s) that sent a zero stops transmitting. The station that sent a 1 continues as normal.

The I.430 and T1.605 standards provide a physical layer protocol that allows multiple terminals to share one physical connection. These protocols handle collisions efficiently because one station can always complete its transmission, at which point, it lowers its own priority to give other devices fair access to the physical connection.

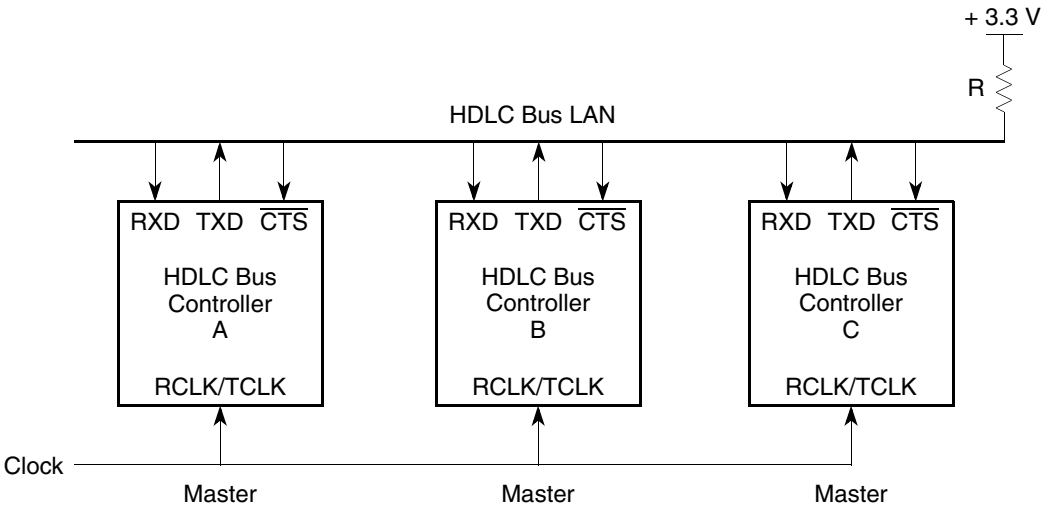
The HDLC bus differs from the I.430 and T1.605 standards as follows:

- The HDLC bus uses a separate input signal rather than the echo bit to monitor data; the transmitted data is simply connected to the $\overline{\text{CTS}}$ input.
- The HDLC bus is a synchronous, digital open-drain connection for short-distance configurations, rather than the more complex S/T interface.
- Any HDLC-based frame protocol can be used at layer 2, not just LAPD.
- HDLC bus devices wait 8–10 rather than 7–10 bit times before transmitting. (HDLC bus has only one class.)

The collision-detection mechanism supports only:

- NRZ-encoded data
- A common synchronous clock for all receivers and transmitters
- Non-inverted data (GSMR[RINV, TINV] = 0)
- Open-drain connection with no external transceivers

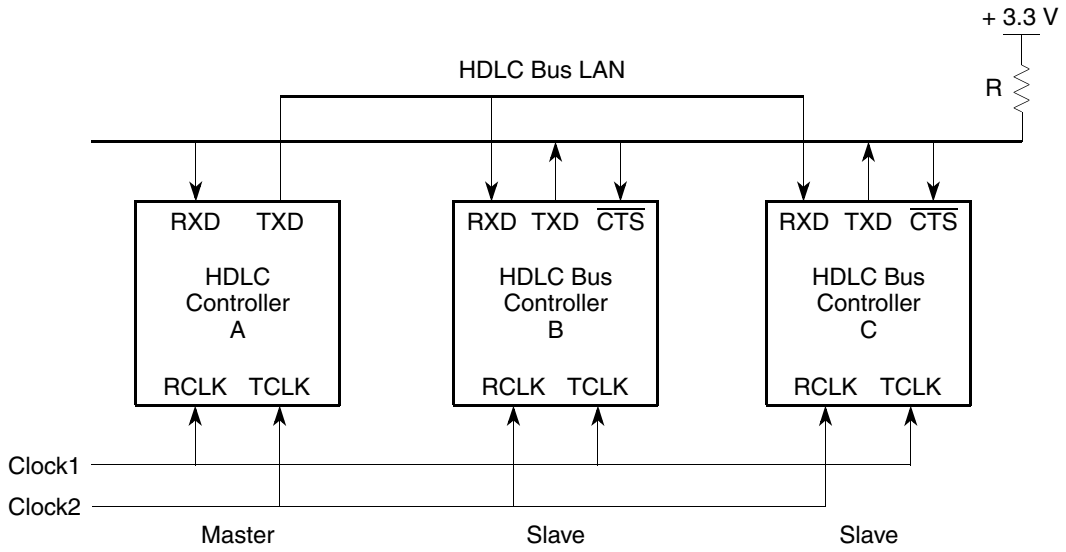
Figure 30-10 shows the most common HDLC bus LAN configuration, a multiple-master configuration. A station can transfer data to or from any other LAN station. Transmissions are half-duplex, which is typical in LANs.



Notes:
1. Transceivers may be used to extend the LAN size.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
3. Clock is a common RCLK/TCLK for all stations.

Figure 30-10. Typical HDLC Bus Multiple-Master Configuration

In single-master configuration, a master station transmits to any slave station without collisions. Slaves communicate only with the master, but can experience collisions in their access over the bus. In this configuration, a slave that communicates with another slave must first transmit its data to the master, where the data is buffered in RAM and then resent to the other slave. The benefit of this configuration, however, is that full-duplex operation can be obtained. In a point-to-multipoint environment, this is the preferred configuration. Figure 30-11 shows the single-master configuration.



Notes:
1. Transceivers may be used to extend the LAN size.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
3. Clock1 is the master RCLK and the slave TCLK.
4. Clock2 is the master TCLK and the slave RCLK.

Figure 30-11. Typical HDLC Bus Single-Master Configuration

30.13.1 HDLC Bus Features

The main features of the HDLC bus are as follows:

- Superset of the HDLC controller features
- Automatic HDLC bus access
- Automatic retransmission in case of collision
- May be used with the NMSI or a TDM bus
- Delayed $\overline{\text{RTS}}$ mode

30.13.2 Accessing the HDLC Bus

The HDLC bus protocol ensures orderly bus control when multiple transmitters attempt simultaneous access. The transmitter sending a zero bit at the time of collision completes the transmission. If a station sends out an opening flag (0x7E) while another station is already sending, the collision is always detected within the first byte, because the transmission in progress is using zero bit insertion to prevent flag imitation.

While in the active condition (ready to transmit), the HDLC bus controller monitors the bus using $\overline{\text{CTS}}$. It counts the one bits on $\overline{\text{CTS}}$. When eight consecutive ones are counted, the HDLC bus controller starts transmitting on the line; if a zero is detected, the internal counter is cleared. During transmission, data is continuously compared with the external bus using $\overline{\text{CTS}}$. $\overline{\text{CTS}}$ is sampled halfway through the bit time using the rising edge of the Tx clock. If the transmitted bit matches the received $\overline{\text{CTS}}$ bus sample, transmission continues. However, if the received $\overline{\text{CTS}}$ sample is 0 and the transmitted bit is 1, transmission stops after that bit and waits for an idle line before attempting retransmission. Since the HDLC bus uses a

wired-OR scheme, a transmitted zero has priority over a transmitted 1. Figure 30-12 shows how $\overline{\text{CTS}}$ is used to detect collisions.

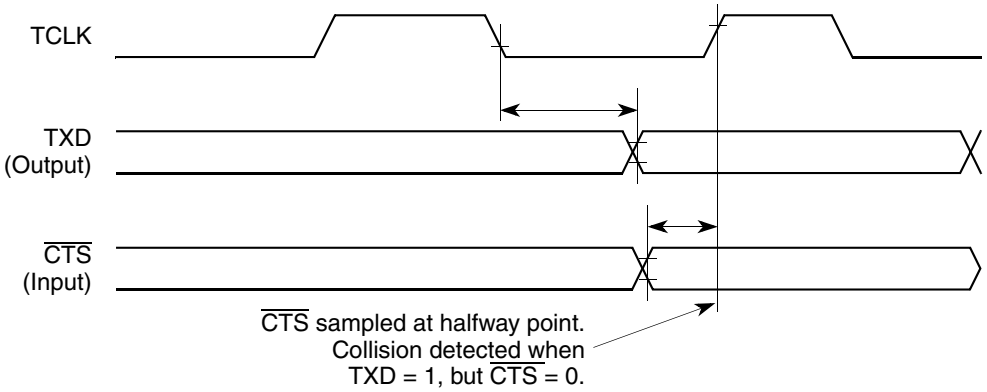


Figure 30-12. Detecting an HDLC Bus Collision

If both the destination address and source address are included in the HDLC frame, then a predefined priority of stations results; if two stations begin to transmit simultaneously, they necessarily detect a collision no later than the end of the source address.

The HDLC bus priority mechanism ensures that stations share the bus equally. To minimize idle time between messages, a station normally waits for eight one bits on the line before attempting transmission. After successfully sending a frame, a station waits for 10 rather than eight consecutive one bits before attempting another transmission. This mechanism ensures that another station waiting to transmit acquires the bus before a station can transmit twice. When a low priority station detects 10 consecutive ones, it tries to transmit; if it fails, it reinstates the high priority of waiting for only eight ones.

30.13.3 Increasing Performance

Because it uses a wired-OR configuration, HDLC bus performance is limited by the rise time of the one bit. To increase performance, give the one bit more rise time by using a clock that is low longer than it is high, as shown in Figure 30-13.

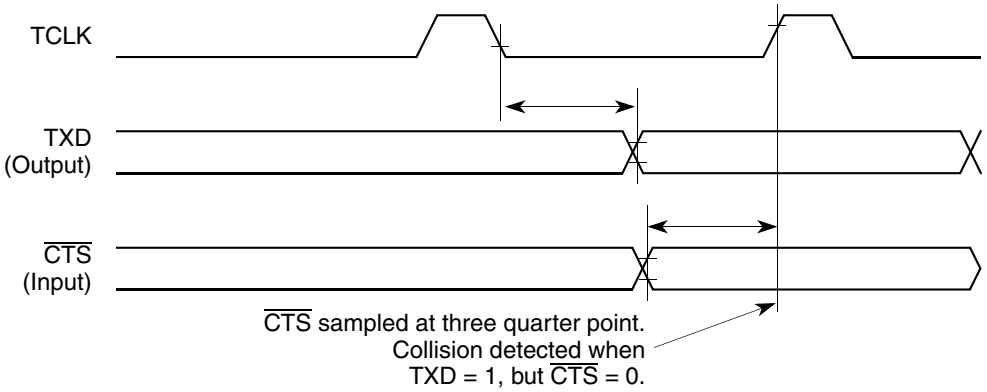
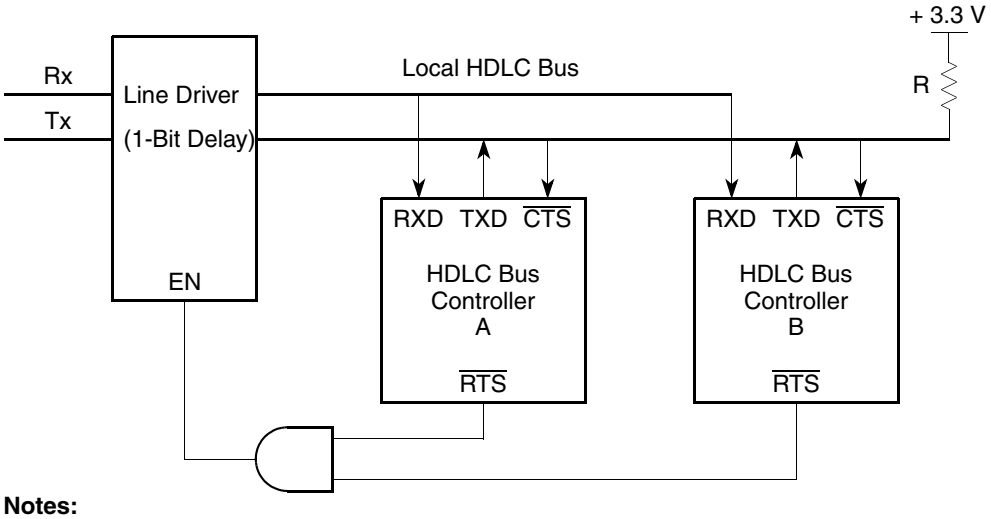


Figure 30-13. Nonsymmetrical Tx Clock Duty Cycle for Increased Performance

30.13.4 Delayed $\overline{\text{RTS}}$ Mode

Figure 30-14 shows local HDLC bus controllers using a standard transmission line and a local bus. The controllers do not communicate with each other but with a station on the transmission line; yet the HDLC bus protocol controls access to the transmission line.



Notes:
1. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
2. The $\overline{\text{RTS}}$ pins of each HDLC bus controller are configured to delayed RTS mode.

Figure 30-14. HDLC Bus Transmission Line Configuration

Normally, $\overline{\text{RTS}}$ goes active at the beginning of the opening flag's first bit. Setting PSMR[BRM] delays $\overline{\text{RTS}}$ by one bit, which is useful when the HDLC bus connects multiple local stations to a transmission line. If the transmission line driver has a one-bit delay, the delayed $\overline{\text{RTS}}$ can be used to enable the output of the line driver. As a result, the electrical effects of collisions are isolated locally. Figure 30-15 shows $\overline{\text{RTS}}$ timing.

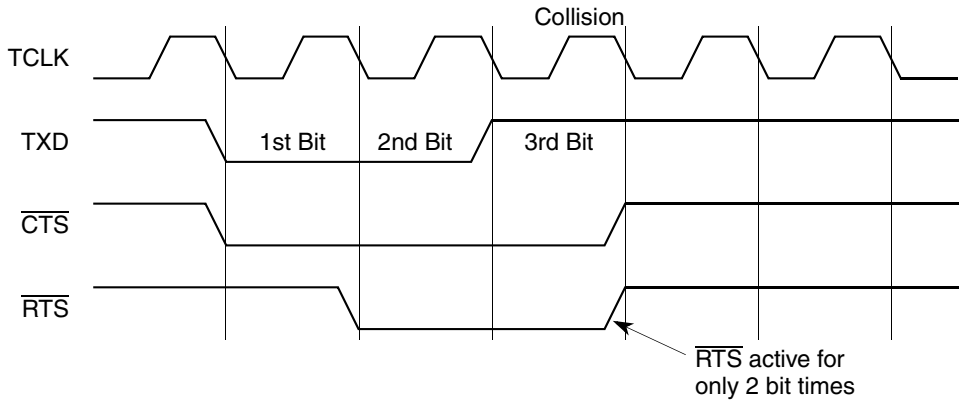
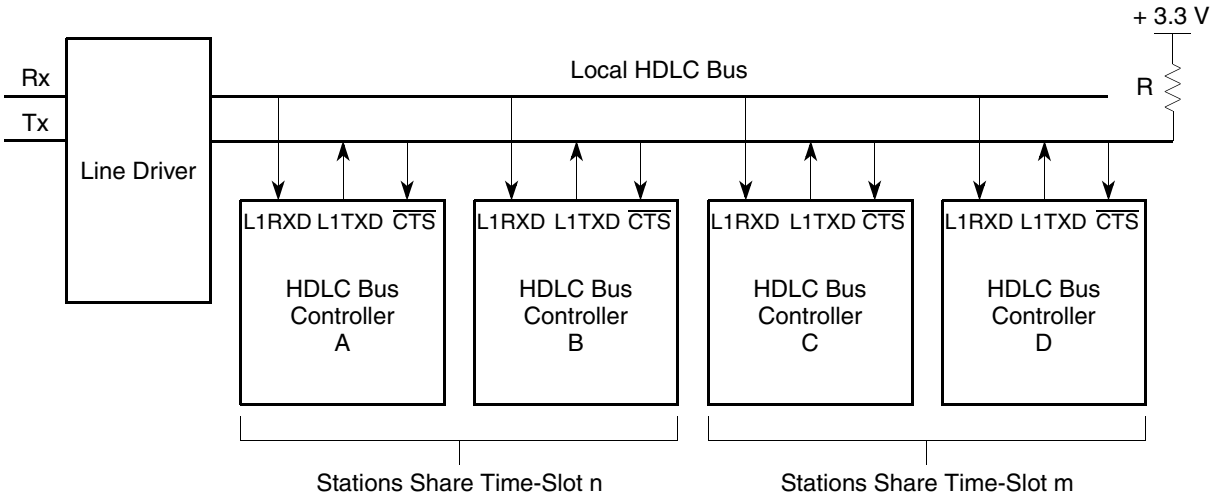


Figure 30-15. Delayed $\overline{\text{RTS}}$ Mode

30.13.5 Using the Time-Slot Assigner (TSA)

HDLC bus controllers can be used with a time-division multiplexed transmission line and a local bus, as shown in Figure 30-16. Local stations use time slots to communicate over the TDM transmission line; stations that share a time slot use the HDLC bus protocol to control access to the local bus.



Notes:
1. All TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
2. The TSA in the SI of each station is used to configure the preferred time slot.
3. The choice of the number of stations to share a time slot is user-defined. It is two in this example.

Figure 30-16. HDLC Bus TDM Transmission Line Configuration

The local SCCs in HDLC bus mode communicate only with the transmission line and not with each other. The SCCs use the TSA of the serial interface, receiving and sending data over L1TXD*x* and L1RXD*x*. Because collisions are still detected from the individual SCC $\overline{\text{CTS}}$ pin, it must be configured to connect to the chosen SCC. Because the SCC only receives clocks during its time slot, $\overline{\text{CTS}}$ is sampled only during the Tx clock edges of the particular SCC time slot.

30.13.6 HDLC Bus Protocol Programming

The HDLC bus on the MPC8555E is implemented using the SCC in HDLC mode with bus-specific options selected in the PSMR and GSMR, as outlined below. See also Section 30.5, “Programming the SCC in HDLC Mode.”

30.13.6.1 Programming GSMR and PSMR for the HDLC Bus Protocol

To program the protocol-specific mode register (PSMR), set the bits as described below:

- Configure NOF as preferred.
- Set RTE and BUS to 1.
- Set BRM to 1 if delayed $\overline{\text{RTS}}$ is desired.
- Configure CRC to 16-bit CRC CCITT (0b00).
- Configure other bits to zero or default.

To program the general SCC mode register (GSMR), set the bits as described below:

- Set MODE to HDLC mode (0b0000).
- Configure CTSS to 1 and all other bits to zero or default.
- Configure the DIAG bits for normal operation (0b00).
- Configure RDCR and TDCR for 1× clock (0b00).
- Configure TENC and RENC for NRZ (0b000).
- Clear RTSM to send idles between frames.
- Set GSMR_L[ENT, ENR] as the last step to begin operation.

Chapter 31
SCC BISYNC Mode

The byte-oriented BISYNC protocol was developed by IBM for use in networking products. The three classes of BISYNC frames, shown in Figure 31-1, are transparent, nontransparent with header, and nontransparent without header. The transparent frame type in BISYNC is not related to transparent mode, discussed in Chapter 32, "SCC Transparent Mode." Transparent BISYNC mode allows full binary data to be sent with any possible character pattern. Each class of frame starts with a standard two-octet synchronization pattern and ends with a block check code (BCC). The end-of-text character (ETX) is used to separate the text and BCC fields.

Nontransparent with Header

SYN1	SYN2	SOH	Header	STX	Text	ETX	BCC

Nontransparent without Header

SYN1	SYN2	STX	Text	ETX	BCC

Transparent

SYN1	SYN2	DLE	STX	Transparent Text	DLE	ETX	BCC

Figure 31-1. Classes of BISYNC Frames

The bulk of a frame is divided into fields whose meaning depends on the frame type. The BCC is a 16-bit CRC format if 8-bit characters are used; it is a combination longitudinal (sum check) and vertical (parity) redundancy check if 7-bit characters are used. In transparent operation, a special character (DLE) is defined that tells the receiver that the next character is text, allowing BISYNC control characters to be valid text data in a frame. A DLE sent as data must be preceded by a DLE character. This is sometimes called byte-stuffing. The physical layer of the BISYNC communications link must synchronize the receiver and transmitter, usually by sending at least one pair of synchronization characters before each frame.

BISYNC protocol is unusual in that a transmit underrun need not be an error. If an underrun occurs, a synchronization pattern is sent until data is again ready. In nontransparent operation, the receiver discards additional synchronization characters (SYNCs) as they are received. In transparent mode, DLE-SYNC pairs are discarded. Normally, for proper transmission, an underrun must not occur between the DLE and its following character. This failure mode cannot occur with the MPC8555E.

An SCC can be configured as a BISYNC controller to handle basic BISYNC protocol in normal and transparent modes. The controller can work with the time-slot assigner (TSA) or nonmultiplexed serial interface (NMSI). The controller has separate transmit and receive sections whose operations are asynchronous with the core and either synchronous or asynchronous with other SCCs.

31.1 Features

The following list summarizes features of the SCC in BISYNC mode:

- Flexible data buffers
- Eight control character recognition registers
- Automatic SYNC1–SYNC2 detection
- 16-bit pattern (bisync)
- 8-bit pattern (monosync)
- 4-bit pattern (nibblesync)
- External SYNC pin support
- SYNC/DLE stripping and insertion
- CRC16 and LRC (sum check) generation/checking
- VRC (parity) generation/checking
- Supports BISYNC transparent operation
- Maintains parity error counter
- Reverse data mode capability

31.2 SCC BISYNC Channel Frame Transmission

The BISYNC transmitter is designed to work with almost no core intervention. When the transmitter is enabled, it starts sending SYN1–SYN2 pairs in the data synchronization register (DSR) or idles as programmed in the PSMR. The BISYNC controller polls the first BD in the channel's TxBD table. If there is a message to send, the controller fetches the message from memory and starts sending it after the SYN1–SYN2 pair. The entire pair is always sent, regardless of GSMR[SYNL].

After a buffer is sent, if the last (TxBD[L]) and the Tx block check sequence (TxBD[TB]) bits are set, the BISYNC controller appends the CRC16/LRC and then writes the message status bits in TxBD status and control fields and clears the ready bit, TxBD[R]. It then starts sending the SYN1–SYN2 pairs or idles, according to GSMR[RTSM]. If the end of the current BD is reached and TxBD[L] is not set, only TxBD[R] is cleared. In both cases, an interrupt is issued according to TxBD[I]. TxBD[I] controls whether interrupts are generated after transmission of each buffer, a specific buffer, or each block. The controller then proceeds to the next BD.

If no additional buffers have been sent to the controller for transmission, an in-frame underrun is detected and the controller starts sending syncs or idles. If the controller is in transparent mode, it sends DLE-sync pairs. Characters are included in the block check sequence (BCS) calculation on a per-buffer basis. Each buffer can be programmed independently to be included or excluded from the BCS calculation; thus, excluded characters must reside in a separate buffer. The controller can reset the BCS generator before sending a specific buffer. In transparent mode, the controller inserts a DLE before sending a DLE character, so that only one DLE is used in the calculation.

31.3 SCC BISYNC Channel Frame Reception

Although the receiver is designed to work with almost no core intervention, the user can intervene on a per-byte basis if necessary. The receiver performs CRC16, longitudinal (LRC) or vertical redundancy (VRC) checking, sync stripping in normal mode, DLE-sync stripping, stripping of the first DLE in DLE-DLE pairs in transparent mode, and control character recognition. Control characters are discussed in Section 31.6, "SCC BISYNC Control Character Recognition."

When enabled, the receiver enters hunt mode where the data is shifted into the receiver shift register one bit at a time and the contents of the shift register are compared to the contents of DSR[SYN1, SYN2]. If the two are unequal, the next bit is shifted in and the comparison is repeated. When registers match, hunt mode is terminated and character assembly begins. The controller is character-synchronized and performs SYNC stripping and message reception. It reverts to hunt mode when it receives an ENTER HUNT MODE command, an error condition, or an appropriate control character.

When receiving data, the controller updates the CR bit in the BD for each byte transferred. When the buffer is full, the controller clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the buffer length, the controller fetches the next BD; if E is zero, reception continues to its buffer.

When a BCS is received, it is checked and written to the buffer. The BISYNC controller sets the last bit, writes the message status bits into the BD, clears the E bit, and then generates a maskable interrupt, indicating that a block of data was received and is in memory. The BCS calculations do not include SYNCs (in nontransparent mode) or DLE-SYNC pairs (in transparent mode).

Note that GSMR_H[RFW] should be set for an 8-bit-wide receive FIFO for the BISYNC receiver. See Section 28.2, "General SCC Mode Registers (GSMR1–GSMR4)."

31.4 SCC BISYNC Parameter RAM

For BISYNC mode, the protocol-specific area of the SCC parameter RAM is mapped as shown in Table 31-1.

Table 31-1. SCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	CRCC	Word	CRC constant temp value
0x38	PRCRC	Hword	Preset receiver/transmitter CRC16/LRC. These values should be preset to all ones or zeros, depending on the BCS used.
0x3A	PTCRC	Hword	
0x3C	PAREC	Hword	Receive parity error counter. This 16-bit (modulo 2^{16}) counter maintained by the CP counts parity errors on receive if the parity feature of BISYNC is enabled. Initialize PAREC while the channel is disabled.
0x3E	BSYNC	Hword	BISYNC SYNC register. Contains the value of the SYNC to be sent as the second byte of a DLE–SYNC pair in an underrun condition and stripped from incoming data on receive once the receiver synchronizes to the data using the DSR and SYN1–SYN2 pair. See Section 31.7, "BISYNC SYNC Register (BSYNC)."

Table 31-1. SCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x40	BDLE	Hword	BISYNC DLE register. Contains the value to be sent as the first byte of a DLE–SYNC pair and stripped on receive. See Section 31.8, "SCC BISYNC DLE Register (BDLE)."
0x42	CHARACTER1	Hword	Control character 1–8. These values represent control characters that the BISYNC controller recognizes. See Section 31.6, "SCC BISYNC Control Character Recognition."
0x44	CHARACTER2	Hword	
0x46	CHARACTER3	Hword	
0x48	CHARACTER4	Hword	
0x4A	CHARACTER5	Hword	
0x4C	CHARACTER6	Hword	
0x4E	CHARACTER7	Hword	
0x50	CHARACTER8	Hword	
0x52	RCCM	Hword	Receive control character mask. Masks CHARACTER*n* comparison so control character classes can be defined. Setting a bit enables and clearing a bit masks comparison. See Section 31.6, "SCC BISYNC Control Character Recognition."

[1] From SCC*x* base address. See Section 28.4.1, "SCC Base Addresses."

GSMR[MODE] determines the protocol for each SCC. The SYN1–SYN2 synchronization characters are programmed in the DSR (see Section 28.2.2, "Data Synchronization Register (DSR).") The BISYNC controller uses the same basic data structure as other modes; receive and transmit errors are reported through their respective BDs. There are two basic ways to handle BISYNC channels:

- The controller can inspect data on a per-byte basis and interrupt the core each time a byte is received.
- The controller can be programmed so software handles the first two or three bytes. The controller directly handles subsequent data without interrupting the core.

31.5 SCC BISYNC Commands

Transmit and receive commands are issued to the CP command register (CPCR). Transmit commands are described in Table 31-2.

Table 31-2. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GSMR, the channel is in transmit enable mode and starts polling the first BD every 64 transmit clocks. This command stops transmission after a maximum of 64 additional bits without waiting for the end of the buffer and the transmit FIFO to be flushed. TBPTR is not advanced, no new BD is accessed, and no new buffers are sent for this channel. SYNC–SYNC or DLE–SYNC pairs are sent continually until a RESTART TRANSMIT is issued. A STOP TRANSMIT can be used when an EOT sequence should be sent and transmission should stop. After transmission resumes, the EOT sequence should be the first buffer sent to the controller. Note that the controller remains in transparent or normal mode after it receives a STOP TRANSMIT or RESTART TRANSMIT command.

Table 31-2. Transmit Commands (continued)

Command	Description
GRACEFUL STOP TRANSMIT	Stops transmission after the current frame finishes sending or immediately if there is no frame being sent. SCCE[GRA] is set once transmission stops. Then BISYNC transmit parameters and TxBDs can be modified. The TBPTR points to the next TxBD. Transmission resumes when the R bit of the next BD is set and a RESTART TRANSMIT is issued.
RESTART TRANSMIT	Lets characters be sent on the transmit channel. The BISYNC controller expects it after a STOP TRANSMIT or a GRACEFUL STOP TRANSMIT command is issued, after a transmitter error occurs, or after a STOP TRANSMIT is issued and the channel is disabled in its SCCM. The controller resumes transmission from the current TBPTR in the channel's TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in the serial channel's parameter RAM to their reset state. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets transmit and receive parameters.

Receive commands are described in Table 31-3.

Table 31-3. Receive Commands

Command	Description
RESET BCS CALCULATION	Immediately resets the receive BCS accumulator. It can be used to reset the BCS after recognizing a control character, thus signifying that a new block is beginning.
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled in SCCM, the channel is in receive enable mode and uses the first BD. This command forces the controller to stop receiving and enter hunt mode, during which the controller continually scans the data stream for an SYN1–SYN2 sequence as programmed in the DSR. After receiving the command, the current receive buffer is closed and the BCS is reset. Message reception continues using the next BD.
CLOSE RXBD	Used to force the SCC to close the current RxBD if it is in use and to use the next BD for subsequent data. If data is not being received, no action is taken.
INIT RX PARAMETERS	Initializes receive parameters in this serial channel's parameter RAM to reset state. Issue only when the receiver is disabled. An INIT TX AND RX PARAMETERS resets transmit and receive parameters.

31.6 SCC BISYNC Control Character Recognition

The BISYNC controller recognizes special control characters that customize the protocol implemented by the BISYNC controller and aid its operation in a DMA-oriented environment. They are used for receive buffers longer than one byte. In single-byte buffers, each byte can be easily inspected so control character recognition should be disabled.

The control character table, shown in Figure 31-2, lets the BISYNC controller recognize the end of the current block. Because the controller imposes no restrictions on the format of BISYNC blocks, software must respond to received characters and inform the controller of mode changes and of certain protocol events, such as resetting the BCS. Using the control character table correctly allows the remainder of the block to be received without interrupting software.

Up to eight control characters can be defined to inform the BISYNC controller that the end of the current block is reached and whether a BCS is expected after the character. For example, the end-of-text character (ETX) implies an end-of-block (ETB) with a subsequent BCS. An enquiry (ENQ) character designates an end of block without a subsequent BCS. All the control characters are written into the data buffer. The BISYNC controller uses a table of 16-bit entries to support control character recognition. Each entry

consists of the control character, an end-of-table bit (E), a BCS expected bit (B), and a hunt mode bit (H). The RCCM entry defines classes of control characters that support masking option.

Offset from SCC*x* Base	0	1	2	3 — 7	8 — 15
0x42	E	B	H	—	CHARACTER1
0x44	E	B	H	—	CHARACTER2
0x46	E	B	H	—	CHARACTER3
0x48	E	B	H	—	CHARACTER4
0x4A	E	B	H	—	CHARACTER5
0x4D	E	B	H	—	CHARACTER6
0x4E	E	B	H	—	CHARACTER7
0x50	E	B	H	—	CHARACTER8
0x52	1	1	1	—	MASK VALUE(RCCM)

Figure 31-2. Control Character Table and RCCM

Table 31-4 describes control character table and RCCM fields.

Table 31-4. Control Character Table and RCCM Field Descriptions

Offset	Bits	Name	Description
0x42–0x50	0	E	End of table. 0 This entry is valid. The lower eight bits are checked against the incoming character. In tables with eight control characters, E should be zero in all eight positions. 1 The entry is not valid. No other valid entries exist beyond this entry.
	1	B	BCS expected. A maskable interrupt is generated after the buffer is closed. 0 The character is written into the receive buffer and the buffer is closed. 1 The character is written into the receive buffer. The receiver waits for one LRC or two CRC bytes of BCS and then closes the buffer. This should be used for ETB, ETX, and ITB.
	2	H	Hunt mode. Enables hunt mode when the current buffer is closed. 0 The BISYNC controller maintains character synchronization after closing this buffer. 1 The BISYNC controller enters hunt mode after closing the buffer. When the B bit is set, the controller enters hunt mode after receiving the BCS.
	3–7	—	Reserved
	8–15	CHARACTER*n*	Control character 1–8. When using 7-bit characters with parity, include the parity bit in the character value.
0x52	0–2	—	All ones
	3–7	—	Reserved
	8–15	RCCM	Received control character mask. Masks comparison of CHARACTER*n*. Each bit of RCCM masks the corresponding bit of CHARACTER*n*. 0 Mask this bit in the comparison of the incoming character and CHARACTER*n*. 1 The address comparison on this bit proceeds normally and no masking occurs. If RCCM is not set, erratic operation can occur during control character recognition.

31.7 BISYNC SYNC Register (BSYNC)

The BSYNC register, as seen in Figure 31-3, defines BISYNC stripping and SYNC character insertion. When an underrun occurs, the BISYNC controller inserts SYNC characters until the next buffer is available for transmission. If the receiver is not in hunt mode when a SYNC character is received, it discards this character if the valid bit (BSYNC[V]) is set. When using 7-bit characters with parity, the parity bit should be included in the SYNC register value.

	0	1	2	3	4	5	6	7	8 15
Field	V	DIS	0	0	0	0	0	0	SYNC
Reset	Undefined								
R/W	R/W								
Offset	SCC Base + 0x3E								

Figure 31-3. BISYNC SYNC (BSYNC)

Table 31-5 describes BSYNC fields.

Table 31-5. BSYNC Field Descriptions

Bits	Name	Description
0	V	Valid. If V = 1 and the receiver is not in hunt mode when a SYNC character is received, this character is discarded.
1	DIS	Disable BSYNC stripping 0 Normal mode 1 BSYNC stripping disabled (BISYNC transparent mode only)
2–7	—	All zeros
8–15	SYNC	SYNC character

31.8 SCC BISYNC DLE Register (BDLE)

The BDLE register, as seen in Figure 31-4, is used to define the BISYNC stripping and insertion of DLE characters. When an underrun occurs while a message is being sent in transparent mode, the BISYNC controller inserts DLE-SYNC pairs until the next buffer is available for transmission.

In transparent mode, the receiver discards any DLE character received and excludes it from the BCS if the valid bit (BDLE[V]) is set. If the second character is SYNC, the controller discards it and excludes it from the BCS. If it is a DLE, the controller writes it to the buffer and includes it in the BCS. If it is not a DLE or SYNC, the controller examines the control character table and acts accordingly. If the character is not in the table, the buffer is closed with the DLE follow character error bit set. If the valid bit is not set, the receiver treats the character as a normal character. When using 7-bit characters with parity, the parity bit should be included in the DLE register value.

	0	1	2	3	4	5	6	7	8	15
Field	V	DIS	0	0	0	0	0	0	DLE	
Reset	Undefined									
R/W	R/W									
Offset	SCC Base + 0x40									

Figure 31-4. BISYNC DLE (BDLE)

Table 31-6 describes BDLE fields.

Table 31-6. BDLE Field Descriptions

Bits	Name	Description
0	V	Valid. If V = 1 and the receiver is not in hunt mode when a SYNC character is received, this character is discarded.
1	DIS	Disable DLE stripping 0 Normal mode 1 DLE stripping disabled. When DIS is enabled in BDLE and on BSYNC the following cases occur: DLE-DLE sequence. Both characters are written to the memory. The BCS is calculated only on the second DLE. DLE-SYNC sequence. Both characters are written to the memory, but neither are included in the BCS calculation. DLE-ETX, DLE-ITB, DLE-ETB sequence, both characters are written to memory. The BCS is calculated only on the second character.
2–7	—	All zeros
8–15	DLE	DLE character

31.9 Sending and Receiving the Synchronization Sequence

The BISYNC channel can be programmed to send and receive a synchronization pattern defined in the DSR. GSMR_H[SYNL] defines pattern length, as shown in Table 31-7. The receiver synchronizes on this pattern. Unless SYNL is zero (external sync), the transmitter always sends the entire DSR contents, lsb first, before each frame—the chosen 4- or 8-bit pattern can be repeated in the lower-order bits.

Table 31-7. Receiver SYNC Pattern Lengths of the DSR

GSMR_H[SYNL] Setting	Bit Assignments															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	An external SYNC signal is used instead of the SYNC pattern in the DSR.															
01	4-bit															
10	8-bit															
11	16-bit															

31.10 Handling Errors in the SCC BISYNC

The controller reports message transmit and receive errors using the channel BDs, error counters, and the SCCE. Modem lines can be directly monitored through the parallel port pins. Table 31-8 describes transmit errors.

Table 31-8. Transmit Errors

Error	Description
Transmitter Underrun	The channel stops sending the buffer, closes it, sets TxBD[UN], and generates aTXE interrupt if it is enabled. The channel resumes transmission after a RESTART TRANSMIT command is received. Underrun cannot occur between frames or during a DLE–*XXX* pair in transparent mode.
$\overline{\text{CTS}}$ Lost during Message Transmission	The channel stops sending the buffer, closes it, sets TxBD[CT], and generates a TXE interrupt if not masked. Transmission resumes when a RESTART TRANSMIT command is received.

Table 31-9 describes receive errors.

Table 31-9. Receive Errors

Error	Description
Overrun	The controller maintains a receiver FIFO for receiving data. The CP begins programming the SDMA channel (if the buffer is in external memory) and updating the CRC when the first byte is received in the Rx FIFO. If an Rx FIFO overrun occurs, the controller writes the received byte over the previously received byte. The previous character and its status bits are lost. The channel then closes the buffer, sets RxBD[OV], and generates the RXB interrupt if it is enabled. Finally, the receiver enters hunt mode.
$\overline{\text{CD}}$ Lost during Message Reception	The channel stops receiving, closes the buffer, sets RxBD[CD], and generates the RXB interrupt if not masked. This error has the highest priority. If the rest of the message is lost, no other errors are checked in the message. The receiver immediately enters hunt mode.
Parity	The channel writes the received character to the buffer and sets RxBD[PR]. The channel stops receiving, closes the buffer, sets RxBD[PR], and generates the RXB interrupt if it is enabled. The channel also increments PAREC and the receiver immediately enters hunt mode.
CRC	The channel updates the CR bit in the BD every time a character is received with a byte delay of 8 serial clocks between the status update and the CRC calculation. When control character recognition is used to detect the end of the block and cause CRC checking, the channel closes the buffer, sets the CR bit in the BD, and generates the RXB interrupt if it is enabled.

31.11 BISYNC Mode Register (PSMR)

The PSMR is used as the BISYNC mode register, shown in Figure 31-5. PSMR[RBCS, RTR, RPM, TPM] can be modified on-the-fly.

	0–3	4–5	6	7	8	9	10–11	12–13	14–15
Field	NOS	CRC	RBCS	RTR	RVD	DRT	—	RPM	TPM
Reset	0								
R/W	R/W								
Offset	0x9_1A08 (PSMR1); 0x9_1A48 (PSMR3); 0x9_1A68 (PSMR4)								

Figure 31-5. Protocol-Specific Mode Register for BISYNC (PSMR)

Table 31-10 describes PSMR fields.

Table 31-10. PSMR Field Descriptions

Bits	Name	Description
0–3	NOS	Minimum number of SYN1–SYN2 pairs (defined in DSR) sent between or before messages.If NOS = 0000, one pair is sent. If NOS = 1111, 16 pairs are sent. The entire pair is always sent, regardless of how GSMR[SYNL] is set. NOS can be modified on-the-fly.
4–5	CRC	CRC selection x0 Reserved. 01 CRC16 (BISYNC). X16 + X15 + X2 + 1. PRCRC and PTCRC should be initialized to all zeros or all ones before the channel is enabled. In either case, the transmitter sends the calculated CRC noninverted and the receiver checks the CRC against zero. Eight-bit data characters (without parity) are configured when CRC16 is chosen. 11 LRC (sum check). (BISYNC). For even LRC, initialize PRCRC and PTCRC to zeros before the channel is enabled; for odd LRC, they should be initialized to ones. **Note:** The receiver checks character parity when BCS is programmed to LRC and the receiver is not in transparent mode. The transmitter sends character parity when BCS is programmed to LRC and the transmitter is not in transparent mode. Use of parity in BISYNC assumes that 7-bit data characters are being used.
6	RBCS	Receive BCS. The receiver internally stores two BCS calculations separated by an eight serial clock delay to allow examination of a received byte to determine whether it should used in BCS calculation. 0 Disable receive BCS 1 Enable receive BCS. Should be set (or reset) within the time taken to receive the following data byte. When RBCS is reset, BCS calculations exclude the latest fully received data byte. When RBCS is set, BCS calculations continue as normal.
7	RTR	Receiver transparent mode 0 Normal receiver mode with SYNC stripping and control character recognition 1 Transparent receiver mode. SYNCs, DLEs, and control characters are recognized only after a leading DLE character. The receiver calculates the CRC16 sequence even if it is programmed to LRC while in transparent mode. Initialize PRCRC to the CRC16 preset value before setting RTR.
8	RVD	Reverse data 0 Normal operation 1 Any portion of this SCC defined to operate in BISYNC mode operates by reversing the character bit order and sending the msb first.
9	DRT	Disable receiver while sending. DRT should not be set for typical BISYNC operation. 0 Normal operation 1 As the SCC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$ signal. This helps if the BISYNC channel is being configured onto a multidrop line and the user does not want to receive its own transmission. Although BISYNC usually uses a half-duplex protocol, the receiver is not actually disabled during transmission.
10–11	—	Reserved, should be cleared.

Table 31-10. PSMR Field Descriptions (continued)

Bits	Name	Description
12–13	RPM	Receiver parity mode. Selects the type of parity check that the receiver performs. RPM can be modified on-the-fly and is ignored unless CRC = 11 (LRC). Receive parity errors cannot be disabled but can be ignored. 00 Odd parity. The transmitter counts ones in the data word. If the sum is not odd, the parity bit is set to ensure an odd number. An even sum indicates a transmission error. 01 Low parity. If the parity bit is not low, a parity error is reported. 10 Even parity. An even number must result from the calculation performed at both ends of the line. 11 High parity. If the parity bit is not high, a parity error is reported.
14–15	TPM	Transmitter parity mode. Selects the type of parity the transmitter performs and can be modified on-the-fly. TPM is ignored unless CRC = 11 (LRC). 00 Odd parity 01 Force low parity (always send a zero in the parity bit position) 10 Even parity 11 Force high parity (always send a one in the parity bit position)

31.12 SCC BISYNC Receive BD (RxBD)

The CP uses BDs to report on each buffer received. It closes the buffer, generates a maskable interrupt, and starts receiving data into the next buffer after any of the following:

- A user-defined control character is received.
- An error is detected.
- A full receive buffer is detected.
- The ENTER HUNT MODE command is issued.
- The CLOSE RX BD command is issued.

Figure 31-6 shows the SCC BISYNC RxBD.

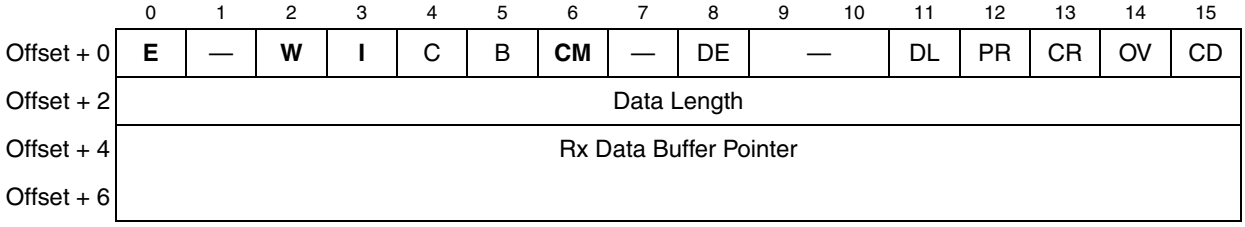


Figure 31-6. SCC BISYNC RxBD

Table 31-11 describes SCC BISYNC RxBD status and control fields.

Table 31-11. SCC BISYNC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty 0 The buffer is full or stopped receiving because of an error. The core can read or write any fields of this RxBD. The CP does not use this BD as long as the E bit is zero. 1 The buffer is not full. The CP controls this BD and buffer. The core should not update this BD.
1	—	Reserved, should be cleared.

Table 31-11. SCC BISYNC RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
2	W	Wrap (last BD in table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to. The number of BDs in this table is determined by the W bit and by overall space constraints of the dual-port RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is used. 1 SCCE[RXB] is set when the controller closes this buffer, which can cause an interrupt if it is enabled.
4	C	Control character. The last byte in the buffer is a user-defined control character. 0 The last byte of this buffer does not contain a control character. 1 The last byte of this buffer contains a control character.
5	B	BCS received. The last bytes in the buffer contain the received BCS. 0 This buffer does not contain the BCS. 1 This buffer contains the BCS. A control character may also reside one byte prior to this BCS.
6	CM	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed; the buffer is overwritten when the CP accesses this BD next. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7	—	Reserved, should be cleared.
8	DE	DPLL error. Set when a DPLL error occurs during reception. In decoding modes where a transition is should occur every bit, the DPLL error is set when a transition is missing.
9–10	—	Reserved, should be cleared.
11	DL	DLE follow character error. While in transparent mode, a DLE character was received, and the next character was not DLE, SYNC, or a valid entry in the control characters table.
12	PR	Parity error. Set when a character with parity error is received. Upon a parity error, the buffer is closed; thus, the corrupted character is the last byte of the buffer. A new Rx buffer receives subsequent data.
13	CR	BCS error. Updated every time a byte is written to the buffer. The CR bit includes the calculation for the current byte. By clearing PSMR[RCBS] within 8 serial clocks, the user can exclude the current character from the message BCS calculation. The data length field may be read to determine the current character's position.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception
15	CD	Carrier detect lost. Indicates when the carrier detect signal, $\overline{CD}$, is negated during frame reception

Data length and buffer pointer fields are described in Section 28.3, "SCC Buffer Descriptors (BDs)." Data length represents the number of octets the CP writes into this buffer, including the BCS. For BISYNC mode, clear these bits. It is incremented each time a received character is written to the buffer.

31.13 SCC BISYNC Transmit BD (TxBD)

The CP arranges data to be sent on an SCC channel in buffers referenced by the channel TxBD table. The CP uses BDs to confirm transmission or indicate errors so the core knows buffers have been serviced. The user configures status and control bits before transmission, but the CP sets them after the buffer is sent.

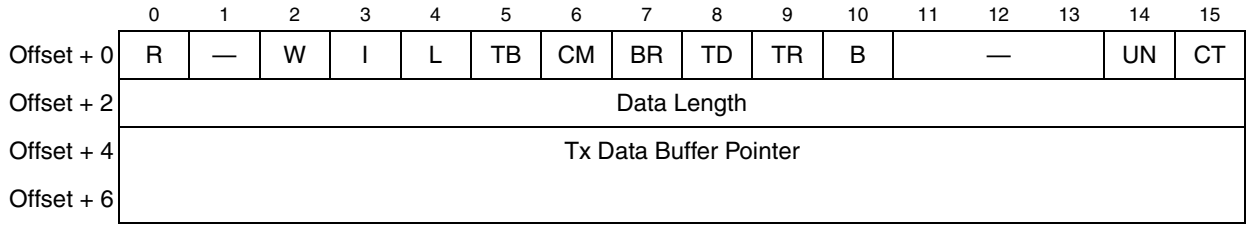


Figure 31-7. SCC BISYNC Transmit BD (TxBD)

Table 31-12 describes SCC BISYNC TxBD status and control fields.

Table 31-12. SCC BISYNC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer is not ready for transmission. The current BD and buffer can be updated. The CP clears R after the buffer is sent or after an error condition. 1 The user-prepared buffer has not been sent or is being sent. This BD cannot be updated while R = 1.
1	—	Reserved, should be cleared.
2	W	Wrap (last BD in table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined only by the W bit and overall space constraints of the dual-ported RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 SCCE[TXB] or SCCE[TXE] is set after the CP services this buffer, which can cause an interrupt.
4	L	Last in message 0 The last character in the buffer is not the last character in the current block. 1 The last character in the buffer is the last character in the current block. The transmitter enters and stays in normal mode after sending the last character in the buffer and the BCS, if enabled.
5	TB	Transmit BCS. Valid only when the L bit is set. 0 Send an SYN1–SYN2 or idle sequence (specified in GSMR[RTSM]) after the last character in the buffer. 1 Send the BCS sequence after the last character. The controller also resets the BCS generator after sending the BCS.
6	CM	Continuous mode 0 Normal operation 1 The CP does not clear R after this BD is closed, so the buffer is resent when the CP next accesses this BD. However, R is cleared if an error occurs during transmission, regardless of how CM is set.
7	BR	BCS reset. Determines whether transmitter BCS accumulation is reset before sending the data buffer. 0 BCS accumulation is not reset. 1 BCS accumulation is reset before sending the data buffer.
8	TD	Transmit DLE 0 No automatic DLE transmission can occur before the data buffer. 1 The transmitter sends a DLE character before sending the buffer, which saves writing the first DLE to a separate buffer in transparent mode. See TR for information on control characters.

Table 31-12. SCC BISYNC TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
9	TR	Transparent mode 0 The transmitter enters and stays in normal mode after sending the buffer. The transmitter automatically inserts SYNCs if an underrun condition occurs. 1 The transmitter enters or stays in transparent mode after sending the buffer. It automatically inserts DLE–SYNC pairs if an underrun occurs (the controller finishes a buffer with L = 0 and the next BD is not available). It also checks all characters before sending them. If a DLE is detected, another DLE is sent automatically. Insert a DLE or program the controller to insert one before each control character. The transmitter calculates the CRC16 BCS even if PSMR[BCS] is programmed to LRC. Initialize PTCRC to CRC16 before setting TR.
10	B	BCS enable 0 The buffer consists of characters that are excluded from BCS accumulation. 1 The buffer consists of characters that are included in BCS accumulation.
11–13	—	Reserved, should be cleared.
14	UN	Underrun. Set when the BISYNC controller encounters a transmitter underrun error while sending the associated data buffer. The CPM writes UN after it sends the associated buffer.
15	CT	$\overline{\text{CTS}}$ lost. The CP sets CT when $\overline{\text{CTS}}$ is lost during message transmission after it sends the data buffer.

Data length and buffer pointer fields are described in Section 28.3, “SCC Buffer Descriptors (BDs).” Although it is never modified by the CP, data length should be greater than zero. The CPM writes these fields after it finishes sending the buffer.

31.14 BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)

The BISYNC controller uses the SCC event register (SCCE) to report events recognized by the BISYNC channel and to generate interrupts. When an event is recognized, the controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the BISYNC mask register (SCCM). SCCE bits are reset by writing ones; writing zeros has no effect. Unmasked bits must be reset before the CP negates the internal interrupt request signal.

	0–4	5	6–7	8	9–10	11	12	13	14	15
Field	—	DCC	—	GRA	—	TXE	RCH	BSY	TXB	RXB
Reset	0000_0000_0000_0000									
R/W	R/W									
Offset	0x9_1A10 (SCCE1); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)									

Figure 31-8. BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)

Table 31-13 describes SCCE and SCCM fields.

Table 31-13. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	DCC	DPLL CS changed. Set when carrier sense status generated by the DPLL changes. Real-time status can be found in SCCS. This is not the $\overline{CD}$ status discussed elsewhere. Valid only when DPLL is used.
6–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. Set as soon the transmitter finishes any message in progress when a GRACEFUL STOP TRANSMIT is issued (immediately if no message is in progress).
9–10	—	Reserved, should be cleared.
11	TXE	Tx Error. Set when an error occurs on the transmitter channel
12	RCH	Receive character. Set when a character is received and written to the buffer
13	BSY	Busy. Set when a character is received and discarded due to a lack of buffers. The receiver resumes reception after an ENTER HUNT MODE command.
14	TXB	Tx buffer. Set when a buffer is sent. TXB is set as the last bit of data or the BCS begins transmission.
15	RXB	Rx buffer. Set when the CPM closes the receive buffer on the BISYNC channel

31.15 SCC Status Registers (SCCS)

The SCC status (SCCS) register, as seen in Figure 31-9, allows real-time monitoring of RXD. The real-time status of $\overline{CTS}$ and $\overline{CD}$ are part of the parallel I/O.

	0	1	2	3	4	5	6	7
Field	—						CS	—
Reset	0000_0000							
R/W	R							
Offset	0x9_1A17 (SCCS1); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)							

Figure 31-9. SCC Status Registers (SCCS)

Table 31-14 describes SCCS fields.

Table 31-14. SCCS Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	—	Reserved, should be cleared.

31.16 Programming the SCC BISYNC Controller

Software has two ways to handle data received by the BISYNC controller. The simplest is to allocate single-byte receive buffers, request an interrupt on reception of each buffer, and implement BISYNC protocol entirely in software on a byte-by-byte basis. This flexible approach can be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is to prepare and link multi-byte buffers in the RxBD table and use software to analyze the first two to three bytes of the buffer to determine the type of block received. When this is determined, reception continues without further software intervention until it encounters a control character, which signifies the end of the block and causes software to revert to byte-by-byte reception.

To accomplish this, set SCCM[RCH] to enable an interrupt on every received byte so software can analyze each byte. After analyzing the initial characters of a block, either set PSMR[RTR] or issue a RESET BCS CALCULATION command. For example, if a DLE-STX is received, enter transparent mode. By setting the appropriate PSMR bit, the controller strips the leading DLE from DLE-character sequences. Thus, control characters are recognized only when they follow a DLE character. PSMR[RTR] should be cleared after a DLE-ETX is received.

Alternatively, after an SOH is received, a RESET BCS CALCULATION should be issued to exclude SOH from BCS accumulation and reset the BCS. Notice that PSMR[RBCS] is not needed because the controller automatically excludes SYNCs and leading DLEs.

After the type of block is recognized, SCCE[RCH] should be masked. The core does not interrupt data reception until the end of the current block, which is indicated by the reception of a control character matching the one in the receive control character table. Using Table 31-15, the control character table should be set to recognize the end of the block.

Table 31-15. Control Characters

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1
ENQ	0	0	0
Next entry	0	X	X

After ETX, a BCS is expected; then the buffer should be closed. HUNT MODE should be entered when a line turnaround occurs. ENQ characters are used to stop sending a block and to designate the end of the block for a receiver, but no CRC is expected. After control character reception, set SCCM[RCH] to re-enable interrupts for each byte of data received.

Chapter 32
SCC Transparent Mode

Transparent mode (also called totally transparent or promiscuous mode) provides a clear channel on which an SCC can send or receive serial data without bit-level manipulation. Software implements protocols run over transparent mode. An SCC in transparent mode functions as a high-speed serial-to-parallel and parallel-to-serial converter.

Transparent mode can be used for serially moving data that requires no superimposed protocol, for applications that require serial-to-parallel and parallel-to-serial conversion for communication among chips on the same board, and for applications that require data to be switched without interfering with the protocol encoding itself, such as when data from a high-speed time-multiplexed serial stream is multiplexed into low-speed data streams. The concept is to switch the data path without altering the protocol encoded on that data path.

Transparent mode is configured in the GSMR; see Section 28.2, “General SCC Mode Registers (GSMR1–GSMR4).” Transparent mode is selected in GSMR_H[TTX, TRX] for the transmitter and receiver, respectively. Setting both bits enables full-duplex transparent operation. If only one is set, the other half of the SCC uses the protocol specified in GSMR_L[MODE]. This allows loopback modes to DMA data from one memory location to another while data is converted to a specific serial format.

The SCC operations are asynchronous with the core and can be synchronous or asynchronous with other SCCs. Each clock can be supplied from the internal baud rate generator bank, DPLL output, or external pins.

The SCC can work with the time-slot assigner (TSA) or nonmultiplexed serial interface (NMSI) and supports modem lines with the general-purpose I/O pins. Data can be transferred either the msb or lsb first in each octet.

32.1 Features

The following list summarizes the main features of the SCC in transparent mode:

- Flexible buffers
- Automatic SYNC detection on receive
- CRCs can be sent and received
- Reverse data mode
- Another protocol can be performed on the other half of the SCC.
- MC68360-compatible SYNC options

32.2 SCC Transparent Channel Frame Transmission Process

The transparent transmitter is designed to work with minimal intervention from the core. When the core enables the SCC transmitter in transparent mode, it starts sending idles, which are logic high or encoded ones, as programmed in GSMR_L[TEND]. The SCC polls the first BD in the TxBD table. When there is a message to send, the SCC fetches data from memory, loads the transmit FIFO, and waits for transmitter synchronization, which is achieved with $\overline{\text{CTS}}$ or by waiting for the receiver to achieve synchronization, depending on GSMR_H[TXSY]. Transmission begins when transmitter synchronization is achieved.

When all BD data has been sent, if TxBD[L] is set, the SCC writes the message status bits into the BD, clears TxBD[R], and sends idles until the next BD is ready. If it is ready, some idles are still sent. The transmitter resumes sending only after it achieves synchronization.

If TxBD[L] is cleared when the end of the BD is reached, only TxBD[R] is cleared and the transmitter moves immediately to the next buffer to begin transmission with no gap on the serial line between buffers. Failure to provide the next buffer in time causes a transmit underrun which sets SCCE[TXE].

In both cases, an interrupt is issued according to TxBD[I]. By appropriately setting TxBD[I] in each BD, interrupts are generated after each buffer or group of buffers is sent. The SCC then proceeds to the next BD in the table and any whole number of bytes can be sent. If GSMR_H[REVD] is set, the bit order of each byte is reversed before being sent; the msb of each octet is sent first.

Setting GSMR_H[TFL] makes the transmit FIFO smaller and reduces transmitter latency, but it can cause transmitter underruns at higher transmission speeds. An optional CRC, selected in GSMR_H[TCRC], can be appended to each transparent frame if it is enabled in the TxBD.

When the time-slot assigner (TSA) is used with a transparent-mode channel, synchronization is provided by the TSA. There is a start-up delay for the transmitter, but delays will always be some whole number of complete TSA frames. This means that *n*-byte transmit buffers can be mapped directly into *n*-byte time slots in the TSA frames.

32.3 SCC Transparent Channel Frame Reception Process

When the core enables the SCC receiver in transparent mode, it waits to achieve synchronization before data is received. The receiver can be synchronized to the data by a synchronization pulse or SYNC pattern.

After a buffer is full, the SCC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. It moves to the next RxBD in the table and begins moving data to its buffer. If the next buffer is not available, SCCE[BSY] signifies a busy signal that can generate a maskable interrupt. The receiver reverts to hunt mode when an ENTER HUNT MODE command or an error is received. If GSMR_H[REVD] is set, the bit order of each byte is reversed before it is written to memory.

Setting GSMR_H[RFW] reduces receiver latency by making the receive FIFO smaller, which may cause receiver overruns at higher transmission speeds. The receiver always checks the CRC of the received frame, according to GSMR_H[TCRC]. If a CRC is not required, resulting errors can be ignored.

32.4 Achieving Synchronization in Transparent Mode

Once the SCC transmitter is enabled for transparent operation, the TxBD is prepared and the transmit FIFO is preloaded by the SDMA channel, another process must occur before data can be sent. It is called transmit synchronization. Similarly, once the SCC receiver is enabled for transparent operation in the GSMR and the RxBD is made empty for the SCC, receive synchronization must occur before data can be received. An in-line synchronization pattern or an external synchronization signal can provide bit-level control of the synchronization process when sending or receiving.

32.4.1 Synchronization in NMSI Mode

This section describes synchronization in NMSI mode.

32.4.1.1 In-Line Synchronization Pattern

The transparent channel can be programmed to receive a synchronization pattern. This pattern is defined in the data synchronization register, DSR; see Section 28.2.2, “Data Synchronization Register (DSR).” Pattern length is specified in GSMR_H[SYNL], as shown in Figure 32-1. See also Section 28.2, “General SCC Mode Registers (GSMR1–GSMR4).”

Table 32-1. Receiver SYNC Pattern Lengths of the DSR

GSMR_H[SYNL] Setting	Bit Assignments															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	An external SYNC signal is used instead of the SYNC pattern in the DSR															
01	4-Bit															
10	8-Bit															
11	16-Bit															

If a 4-bit SYNC is selected, reception begins as soon as these four bits are received, beginning with the first bit following the 4-bit SYNC. The transmitter synchronizes on the receiver pattern if GSMR_H[RSYN] = 1.

Note that the transparent controller does not automatically send the synchronization pattern; therefore, the synchronization pattern must be included in the transmit buffer.

32.4.1.2 External Synchronization Signals

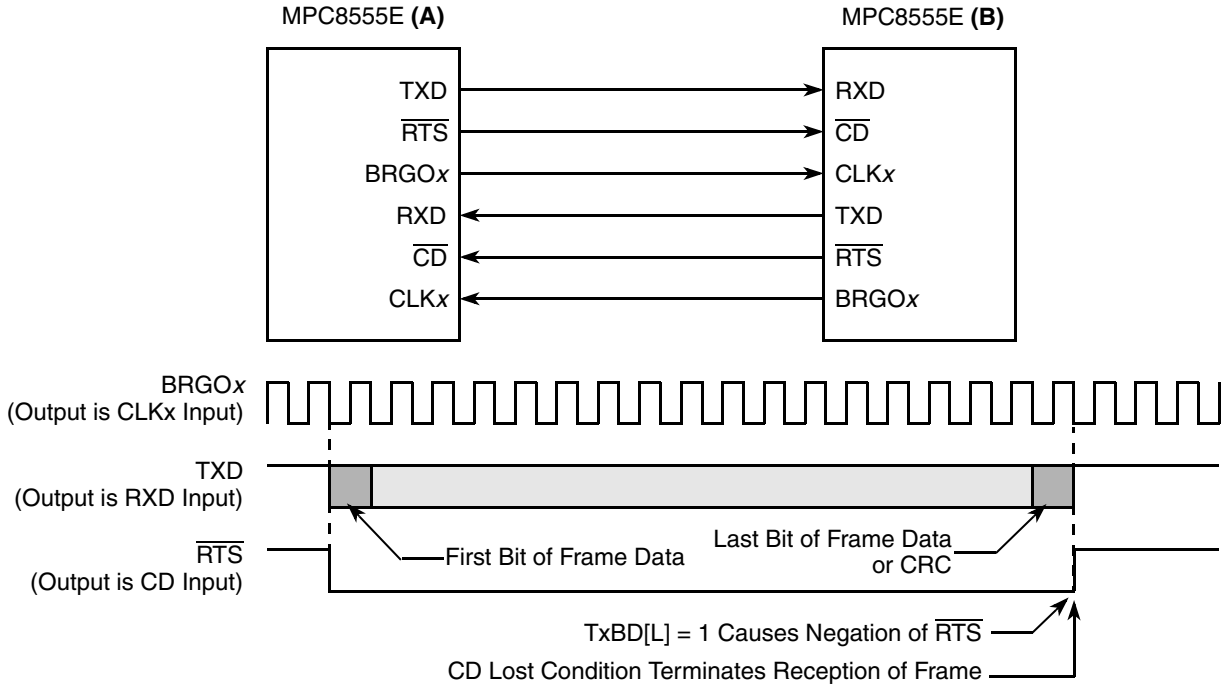
If GSMR_H[SYNL] is 0b00, the transmitter uses $\overline{\text{CTS}}$ and the receiver uses $\overline{\text{CD}}$ to begin the sequence. These signals share two options—pulsing and sampling.

GSMR_H[CDP] and GSMR_H[CTSP] determine whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ need to be asserted only once to begin reception/transmission or whether they must remain asserted for the duration of the transparent frame. Pulse operation allows an uninterrupted stream of data. However, use envelope mode to identify frames of transparent data.

The sampling option determines the delay between $\overline{CD}$ and $\overline{CTS}$ being asserted and the resulting action by the SCC. Assume either that these signals are asynchronous to the data and internally synchronized by the SCC or that they are synchronous to the data with faster operation. This option allows $\overline{RTS}$ of one SCC to be connected to $\overline{CD}$ of another SCC and to have the data synchronized and bit aligned. It is also an option to link the transmitter synchronization to the receiver synchronization. Diagrams for the pulse/envelope and sampling options are shown in Section 32.4, "Achieving Synchronization in Transparent Mode."

32.4.1.2.1 External Synchronization Example

Figure 32-1 shows synchronization using external signals.



Notes:
- Each MPC8555E generates its own transmit clocks. If the transmit and receive clocks are the same, one MPC8555E can generate transmit and receive clocks for the other MPC8555E. For example, CLK*x* on MPC8555E (B) could be used to clock the transmitter and receiver.
- $\overline{CTS}$ should be configured as always asserted in the parallel I/O or connected to ground externally.
- The required GSMR configurations are DIAG = 00, CTSS = 1, CTSP is a 'don't care', CDS = 1, CDP = 0, TTX = 1, and TRX = 1. REVD and TCRC are application-dependent.
- The transparent frame contains a CRC if TxBD[TC] is set.

Figure 32-1. Sending Transparent Frames Between MPC8555Es

MPC8555E(A) and MPC8555E(B) exchange transparent frames and synchronize each other using $\overline{RTS}$ and $\overline{CD}$. However, $\overline{CTS}$ is not required because transmission begins at any time. Thus, $\overline{RTS}$ is connected directly to the other MPC8555E $\overline{CD}$ pin. GSMR_H[RSYN] is not used and transmission and reception from each MPC8555E are independent.

32.4.1.3 Transparent Mode Without Explicit Synchronization

If there is no need to synchronize the transparent controller at a specific point, the user can ‘fake’ synchronization in one of the following ways:

- Tie a parallel I/O pin to the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ lines. Then, after enabling the receiver and transmitter, provide a falling edge by manipulating the I/O pin in software.
- Enable the receiver and transmitter for the SCC in loopback mode and then change GSMR_L[DIAG] to 0b00 while the transmitter and receiver and enabled.

32.4.2 Synchronization and the TSA

A transparent-mode SCC using the time-slot assigner can synchronize either on a user-defined inline pattern or by inherent synchronization.

Note that when using the TSA, a newly-enabled transmitter sends from 10 to 15 frames of idles before sending the actual transparent data due to startup requirements of the TDM. Therefore, when loopback testing through the TDM, expect to receive several bytes of 0xFF before the actual data.

32.4.2.1 Inline Synchronization Pattern

The receiver can be programmed to begin receiving data into the receive buffers only after a specified data pattern arrives. To synchronize on an inline pattern:

- Set GSMR_H[SYNL].
- Program the DSR with the desired pattern.
- Clear GSMR_H[CDP].
- Set GSMR_H[CTSP, CTSS, CDS].

If GSMR_H[TXSY] is also used, the transmitter begins transmission 8 clocks after the receiver achieves synchronization.

32.4.2.2 Inherent Synchronization

Inherent synchronization assumes synchronization by default when the channel is enabled; all data sent from the TDM to the SCC is received. To implement inherent synchronization:

- Set GSMR_H[CDP, CDS, CTSP, CTSS].

If these bits are not set, the received bit stream will be bit-shifted. The SCC loses the first received bit because $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are treated as asynchronous signals.

32.4.3 End of Frame Detection

An end of frame cannot be detected in the transparent data stream since there is no defined closing flag in transparent mode. Therefore, if framing is needed, the user must use the $\overline{\text{CD}}$ line to alert the transparent controller of an end of frame.

32.5 CRC Calculation in Transparent Mode

The CRC calculations follow the ITU/IEEE standard. The CRC is calculated on the transmitted data stream; that is, from lsb to msb for non-bit-reversed (GSMR_H[REVD] = 0) and from msb to lsb for bit-reversed (GSMR_H[REVD] = 1) transmission. The appended CRC is sent msb to lsb.

When receiving, the CRC is calculated as the incoming bits arrive. The optional reversal of data (GSMR_H[REVD] = 1) is done just before data is stored in memory (after the CRC calculation).

32.6 SCC Transparent Parameter RAM

For transparent mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 32-2.

Table 32-2. SCC Transparent Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x 30	CRC_P	Long	CRC preset for totally transparent. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF. For the 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF and for the CRC-16, initialize with ones (0x0000_FFFF) or zeros (0x0000_0000).
0x 34	CRC_C	Long	CRC constant for totally transparent receiver. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8. For the 32-bit CRC-CCITT, CRC_C initialize with 0xDEBB_20E3 and for the CRC-16, which is normally used with BISYNC, initialize with 0x0000_0000.

1 From SCC base address. See Section 28.4.1, “SCC Base Addresses.”

CRC_P and CRC_C overlap with the CRC parameters for the HDLC-based protocols. However, this overlap is not detrimental since the CRC constant is used only for the receiver and the CRC preset is used only for the transmitter, so only one entry is required for each. Thus, the user can choose an HDLC transmitter with a transparent receiver or a transparent transmitter with an HDLC receiver.

32.7 SCC Transparent Commands

The following transmit and receive commands are issued to the CP command register. Table 32-3 describes transmit commands.

Table 32-3. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GSMR, the channel is in transmit enable mode and starts polling the first BD every 64 clocks (or immediately if TODR[TOD] = 1). STOP TRANSMIT disables frame transmission on the transmit channel. If the transparent controller receives the command during frame transmission, transmission is aborted after a maximum of 64 additional bits and the transmit FIFO is flushed. The current TxBD pointer (TBPTR) is not advanced, no new BD is accessed and no new buffers are sent for this channel. The transmitter will send idles.
GRACEFUL STOP TRANSMIT	Stops transmission smoothly, rather than abruptly, in much the same way that the regular STOP TRANSMIT command stops. It stops transmission after the current frame finishes or immediately if no frame is being sent. A transparent frame is not complete until a BD with TxBD[L] set has its buffer completely sent. SCCE[GRA] is set once transmission stops; transmit parameters and their BDs can then be modified. The current TxBD pointer (TBPTR) advances to the next TxBD in the table. Transmission resumes once TxBD[R] is set and a RESTART TRANSMIT command is issued.

Table 32-3. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Re-enables transmission of characters on the transmit channel. The transparent controller expects it after a STOP TRANSMIT command is issued (at which point the channel is disabled in SCCM), after a GRACEFUL STOP TRANSMIT command is issued, or after a transmitter error. The transparent controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in the serial channel parameter RAM to reset state. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

Table 32-4 describes receive commands.

Table 32-4. Receive Commands

Command	Description
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled, the channel is in receive enable mode and uses the first BD in the table. ENTER HUNT MODE forces the transparent receiver to the current frame and enter hunt mode where the transparent controller waits for the synchronization sequence. After receiving the command, the current buffer is closed. Further data reception uses the next BD.
CLOSE RXBD	Forces the SCC to close the RxBD if it is being used and to use the next BD for any subsequently received data. If the SCC is not receiving data, no action is taken by this command.
INIT RX PARAMETERS	Initializes all receive parameters in this serial channel parameter RAM to reset state. Issue only when the receiver is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

32.8 Handling Errors in the Transparent Controller

The SCC reports message reception and transmission errors using the channel buffer descriptors, the error counters, and SCCE. Table 32-5 describes transmit errors.

Table 32-5. Transmit Errors

Error	Description
Transmitter Underrun	When this occurs, the channel stops sending the buffer, closes it, sets TxBD[UN], and generates a TXE interrupt if it is enabled. Transmission resumes after a RESTART TRANSMIT command is received. Underrun occurs after a transmit frame for which TxBD[L] was not set. In this case, only SCCE[TXE] is set. Underrun cannot occur between transparent frames.
$\overline{\text{CTS}}$ Lost During Message Transmission	When this occurs, the channel stops sending the buffer, closes it, sets TxBD[CT], and generates the TXE interrupt if it is enabled. The channel resumes sending after RESTART TRANSMIT is received.

Table 32-6 describes receive errors.

Table 32-6. Receive Errors

Error	Description
Overrun	The SCC maintains a receive FIFO. The CPM starts programming the SDMA channel if the buffer is in external memory and updating the CRC when 8 or 32 bits are received in the FIFO as determined by GSMR_H[RFW]. If a FIFO overrun occurs, the SCC writes the received byte over the previously received byte. The previous character and its status bits are lost. Afterwards, the channel closes the buffer, sets OV in the BD, and generates the RXB interrupt if it is enabled. The receiver immediately enters hunt mode.
$\overline{\text{CD}}$ Lost During Message Reception	When this occurs, the channel stops receiving messages, closes the buffer, sets RxBD[CD], and generates the RXB interrupt if it is enabled. This error has highest priority. The rest of the message is lost, and no other errors are checked in the message. The receiver immediately enters hunt mode.

32.9 Transparent Mode and the PSMR

The protocol-specific mode register (PSMR) is not used by the transparent controller because all transparent mode selections are made in the GSMR. If only half of an SCC (transmitter or receiver) is running the transparent protocol, the other half (receiver or transmitter) can support another protocol. In such a case, use the PSMR for the non-transparent protocol.

32.10 SCC Transparent Receive Buffer Descriptor (RxBD)

The CPM reports information about the received data for each buffer using an RxBD (see Figure 32-2), closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- An error is detected.
- A full receive buffer is detected.
- An ENTER HUNT MODE command is issued.
- A CLOSE RXBD command is issued.

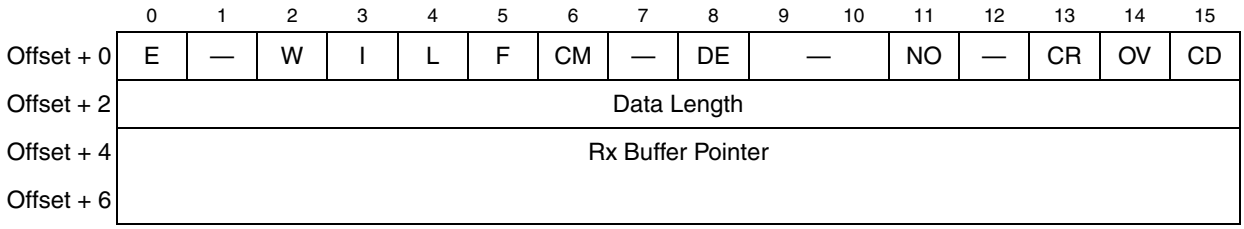


Figure 32-2. SCC Transparent Receive Buffer Descriptor (RxBD)

Table 32-7 describes RxBD status and control fields.

Table 32-7. SCC Transparent RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty 0 The buffer is full or stopped receiving data because an error occurred. The core can read or write to any fields of this RxBD. The CPM does not use this BD when RxBD[E] is zero. 1 The buffer is not full. This RxBD and buffer are owned by the CPM. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM receives data into the first BD that RBASE points to. The number of BDs in this table is determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is used. 1 When this buffer is closed by the transparent controller, the SCCE[RXB] is set. SCCE[RXB] can cause an interrupt if it is enabled.
4	L	Last in frame. Set by the transparent controller when this buffer is the last in a frame, which occurs when $\overline{CD}$ is negated (if GSMR_H[CDP] = 0) or an error is received. If an error is received, one or more of RxBD[OV, CD, DE] are set. Note that the SCC transparent controller writes the number of buffer (not frame) octets to the last BD's data length field. 0 Not the last buffer in a frame 1 Last buffer in a frame
5	F	First in frame. The transparent controller sets F when this buffer is the first in the frame: 0 Not the first buffer in a frame 1 First buffer in a frame
6	CM	Continuous mode 0 Normal operation 1 The CPM does not clear RxBD[E] after this BD is closed, letting the buffer be overwritten when the CPM next accesses this BD. However, RxBD[E] is cleared if an error occurs during reception, regardless of how CM is set.
7	—	Reserved, should be cleared.
8	DE	DPLL error. Set by the transparent controller when a DPLL error occurs as this buffer is received. In decoding modes, where a transition is promised every bit, DE is set when a missing transition occurs. If a DPLL error occurs, no other error checking is performed.
9–10	—	Reserved, should be cleared.
11	NO	Rx non-octet. Set when a frame containing a number of bits not exactly divisible by eight is received.
12	—	Reserved, should be cleared.
13	CR	CRC error indication bits. Indicates that this frame contains a CRC error. The received CRC bytes are always written to the receive buffer. CRC checking cannot be disabled, but it can be ignored.
14	OV	Overrun. Indicates that a receiver overrun occurred during buffer reception
15	CD	Carrier detect lost. Indicates when $\overline{CD}$ is negated during buffer reception

Data length and buffer pointer fields are described in Section 28.3, "SCC Buffer Descriptors (BDs)." The Rx buffer pointer must be divisible by four, unless GSMR_H[RFW] is set to 8 bits wide, in which case the pointer can be even or odd. The buffer can reside in internal or external memory.

32.11 SCC Transparent Transmit Buffer Descriptor (TxBD)

Data is sent to the CPM for transmission on an SCC channel by arranging it in buffers referenced by the TxBD table. The CPM uses BDs to confirm transmission or indicate error conditions so the processor knows buffers have been serviced. Status and control bits must be prepared before transmission; they are set by the CPM after the buffer is sent. The TxBD is shown in Figure 32-3.

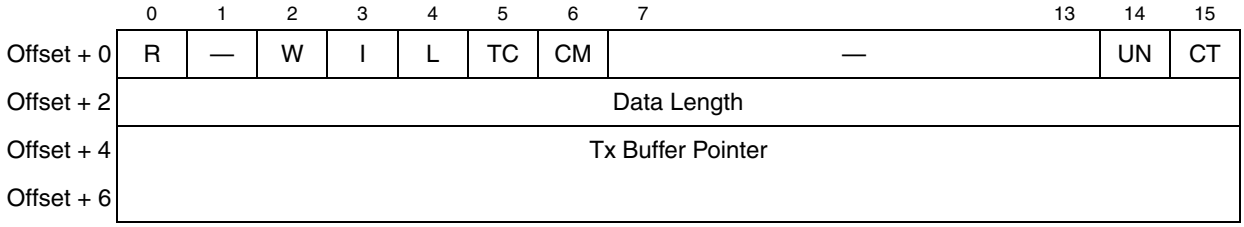


Figure 32-3. SCC Transparent Transmit Buffer Descriptor (TxBD)

Figure 32-8 describes SCC transparent TxBD status and control fields.

Table 32-8. SCC Transparent TxBD Status and Control Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer is not ready for transmission. The BD and buffer can be updated. The CPM clears R after the buffer is sent or after an error is encountered. 1 The user-prepared buffer is not sent yet or is being sent. This BD cannot be updated while R = 1.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is determined only by TxBD[W] and overall space constraints of the dual-port RAM.
3	I	Interrupt. Note that clearing this bit does not disable all SCCE[TXE] events. 0 No interrupt is generated after this buffer is serviced. 1 When the CPM services this buffer, SCCE[TXB] or SCCE[TXE] is set. These bits can cause interrupts if they are enabled.
4	L	Last in message 0 The last byte in the buffer is not the last byte in the transmitted transparent frame. Data from the next transmit buffer is sent immediately after the last byte of this buffer. 1 The last byte in the buffer is the last byte in the transmitted transparent frame. After this buffer is sent, the transmitter requires synchronization before the next buffer is sent.
5	TC	Transmit CRC. 0 No CRC sequence is sent after this buffer. 1 A frame check sequence defined by GSMR_H[TCRC] is sent after the last byte of this buffer.

Table 32-8. SCC Transparent TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
6	CM	Continuous mode 0 Normal operation 1 The CPM does not clear TxBD[R] after this BD is closed, so the buffer is automatically resent when the CPM accesses this BD next. However, TxBD[R] is cleared if an error occurs during transmission, regardless of how CM is set.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. Set when the SCC encounters a transmitter underrun condition while sending the buffer.
15	CT	CTS lost. Indicates the $\overline{\text{CTS}}$ was lost during frame transmission.

Data length and buffer pointer fields are described in Section 28.3, "SCC Buffer Descriptors (BDs)." Although it is never modified by the CP, data length should be greater than zero. The buffer pointer can be even or odd and can reside in internal or external memory.

32.12 SCC Transparent Event Register (SCCE)/Mask Register (SCCM)

When the SCC is in transparent mode, the SCC event register (SCCE) functions as the transparent event register to report events recognized by the transparent channel and to generate interrupts. When an event is recognized, the transparent controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the transparent mask register (SCCM).

Event bits are reset by writing ones; writing zeros has no effect. All unmasked bits must be reset before the CP clears the internal interrupt request to the CPM interrupt controller. Figure 32-4 shows the event and mask registers.

	0–4	5	6–7	8	9–10	11	12	13	14	15
Field	—	DCC	—	GRA	—	TXE	—	BSY	TXB	RXB
Reset	0000_0000_0000_0000									
R/W	R/W									
Offset	0x9_1A10 (SCCE1); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)									

Figure 32-4. SCC Transparent Event Register (SCCE)/Mask Register (SCCM)

Table 32-9 describes SCCE/SCCM fields.

Table 32-9. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	DCC	DPLL CS changed. Set when the DPLL-generated carrier sense status changes (valid only when the DPLL is used). Real-time status can be read in SCCS. This is not the $\overline{\text{CD}}$ status mentioned elsewhere.
6–7	—	Reserved, should be cleared.

Table 32-9. SCCE/SCCM Field Descriptions (continued)

Bits	Name	Description
8	GRA	Graceful stop complete. Set when a graceful stop initiated by completes as soon as the transmitter finishes any frame in progress when the GRACEFUL STOP TRANSMIT command was issued. Immediately if no frame was in progress when the command was issued.
9–10	—	Reserved, should be cleared.
11	TXE	Tx error. Set when an error occurs on the transmitter channel.
12	—	Reserved, should be cleared.
13	BSY	Busy condition. Set when a byte or word is received and discarded due to a lack of buffers. The receiver resumes reception after it gets an ENTER HUNT MODE command.
14	TXB	Tx buffer. Set no sooner than when the last bit of the last byte of the buffer begins transmission, assuming L is set in the TxBD. If it is not, TXB is set when the last byte is written to the transmit FIFO.
15	RXB	Rx buffer. Set when a complete buffer was received on the SCC channel, no sooner than 2 serial clocks after the last bit of the last byte in which the buffer is received on RXD.

32.13 SCC Status Register in Transparent Mode (SCCS)

The SCC status register (SCCS), shown in Figure 32-5, allows monitoring of real-time status conditions on the RXD line. The real-time status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are part of the parallel I/O.

	0 — 5	6	7
Field	—	CS	—
Reset	0000_0000		
R/W	R		
Offset	0x9_1A17 (SCCS1); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)		

Figure 32-5. SCC Status Register in Transparent Mode (SCCS)

Table 32-10 describes SCCS fields.

Table 32-10. SCCS Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	—	Reserved, should be cleared.

Chapter 33
SCC AppleTalk Mode

AppleTalk is a set of protocols developed by Apple Computer, Inc. to provide a LAN service between Macintosh computers and printers. Although AppleTalk can be implemented over a variety of physical and link layers, including Ethernet, AppleTalk protocols have been most closely associated with the LocalTalk physical and link-layer protocol, an HDLC-based protocol that runs at 230.4 Kbps. In this manual, the term 'AppleTalk controller' refers to the support that the MPC8555E provides for LocalTalk protocol. The AppleTalk controller provides required frame synchronization, bit sequence, preamble, and onto standard HDLC frames. These capabilities, with the use of the HDLC controller in conjunction with DPLL operation in FM0 mode, provide the proper connection formats to the LocalTalk bus.

33.1 Operating the LocalTalk Bus

A LocalTalk frame, shown in Figure 33-1, is basically a modified HDLC frame.

Sync Sequence	HDLC Flags	Destination Address	Source Address	Control Byte	Data (Optional)	CRC-16	Closing Flag	Abort Sequence
> 3 Bits	2 or More Bytes	1 Byte	1 Byte	1 Byte	0–600 Bytes	2 Bytes	1 Byte	12–18 Ones

Figure 33-1. LocalTalk Frame Format

First, a synchronization sequence of more than three bits is sent. This sequence consists of at least one logical one bit (FM0 encoded) followed by two bit times or more of line idle with no particular maximum time specified. The idle time allows LocalTalk equipment to sense a carrier by detecting a missing clock on the line. The remainder of the frame is a typical half-duplex HDLC frame. Two or more flags are sent, allowing bit, byte, and frame delineation or detection. Two bytes of address, destination, and source are sent next, followed by a byte of control and 0–600 data bytes. Next, two bytes of CRC (the common 16-bit CRC-CCITT polynomial referenced in the HDLC standard protocol) are sent. The LocalTalk frame is then terminated by a flag and a restricted HDLC abort sequence. Then the transmitter's driver is disabled.

The control byte within the LocalTalk frame indicates the type of frame. Control byte values from 0x01–0x7F are data frames; control byte values from 0x80–0xFF are control frames. Four control frames are defined:

- ENQ—Enquiry
- ACK—Enquiry acknowledgment
- $\overline{\text{RTS}}$—Request to send a data frame
- $\overline{\text{CTS}}$—Clear to send a data frame

Frames are sent in groups known as dialogs, which are handled by the software. For instance, to transfer a data frame, three frames are sent over the network. An $\overline{\text{RTS}}$ frame (not to be confused with the RS-232

$\overline{\text{RTS}}$ pin) is sent to request the network, a $\overline{\text{CTS}}$ frame is sent by the destination node, and the data frame is sent by the requesting node. These three frames comprise one possible type of dialog. After a dialog begins, other nodes cannot start sending until the dialog is complete. Frames within a dialog are sent with a maximum interframe gap (IFG) of 200 µs. Although the LocalTalk specification does not state it, there is also a minimum recommended IFG of 50 µs. Dialogs must be separated by a minimum interdialog gap (IDG) of 400 µs. In general, these gaps are implemented by the software.

Depending on the protocol, collisions should be encountered only during $\overline{\text{RTS}}$ and ENQ frames. Once frame transmission begins, it is fully sent, regardless of whether it collides with another frame. ENQ frames are infrequent and are sent only when a node powers up and enters the network. A higher-level protocol controls the uniqueness and transmission of ENQ frames.

In addition to the frame fields, LocalTalk requires that the frame be FM0 (differential Manchester space) encoded, which requires one level transition on every bit boundary. If the value to be encoded is a logical zero, FM0 requires a second transition in the middle of the bit time. The purpose of FM0 encoding is to avoid having to transmit clocking information on a separate wire. With FM0, the clocking information is present whenever valid data is present.

33.2 Features

The following list summarizes the features of the SCC in AppleTalk mode:

- Superset of the HDLC controller features
- FM0 encoding/decoding
- Programmable transmission of sync sequence
- Automatic postamble transmission
- Reception of sync sequence does not cause extra SCCE[DCC] interrupts
- Reception is automatically disabled while sending a frame
- Transmit-on-demand feature expedites frames
- Connects directly to an RS-422 transceiver

33.3 Connecting to AppleTalk

As shown in Figure 33-2, the MPC8555E connects to LocalTalk, and, using TXD, $\overline{\text{RTS}}$, and RXD, is an interface for the RS-422 transceiver. The RS-422, in turn, is an interface for the LocalTalk connector. Although it is not shown, a passive RC circuit is recommended between the transceiver and connector.

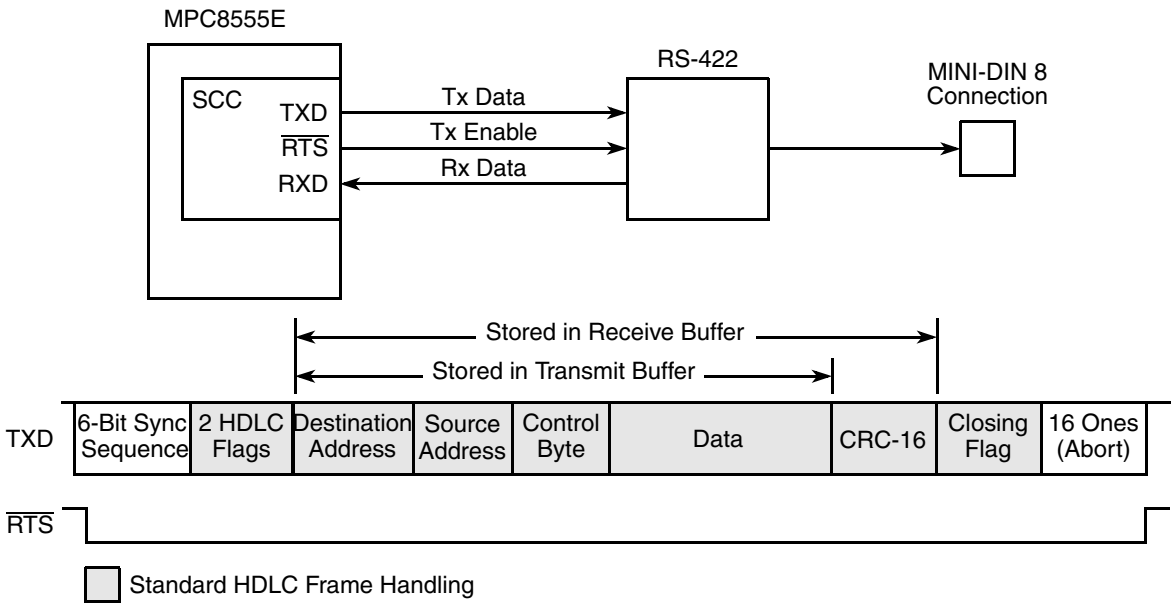


Figure 33-2. Connecting the MPC8555E to LocalTalk

The 16x overspeed of a 3.686-MHz clock can be generated from an external frequency source or from one of the baud rate generators if the resulting output frequency is close to a multiple of the 3.686 MHz frequency. The MPC8555E asserts $\overline{\text{RTS}}$ throughout the duration of the frame so that $\overline{\text{RTS}}$ can be used to enable the RS-422 transmit driver.

33.4 Programming the SCC in AppleTalk Mode

The AppleTalk controller is implemented by setting certain bits in the HDLC controller. Otherwise, Chapter 30, "SCC HDLC Mode," describes how to program the HDLC controller. Use GSMR, PSMR, or TODR to program the AppleTalk controller.

33.4.1 Programming the GSMR

Program the GSMR as described below:

1. Set MODE to 0b0010 (AppleTalk).
2. Set DIAG to 0b00 for normal operation, with $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ grounded or configured for parallel I/O. This causes $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ to be internally asserted to the SCC.
3. Set RDCR and TDCR to (0b10) a 16× clock.
4. Set the TENC and RENC bits to 0b010 (FM0).
5. Clear TEND for default operation.
6. Set TPP to 0b11 for a preamble pattern of all ones.
7. Set TPL to 0b000 to transmit the next frame with no synchronization sequence and to 001 to transmit the next frame with the LocalTalk synchronization sequence. For example, data frames do not require a preceding synchronization sequence. These bits may be modified on-the-fly if the AppleTalk protocol is selected.

8. Clear TINV and RINV so data will not be inverted.
9. Set TSNC to 1.5 bit times (0b10).
10. Clear EDGE. Both the positive and negative edges are used to change the sample point (default).
11. Clear RTSM (default).
12. Set all other bits to zero or default.
13. Set ENT and ENR as the last step to begin operation.

33.4.2 Programming the PSMR

Follow these steps to program the protocol-specific mode register:

1. Set NOF to 0b0001 giving two flags before frames (one opening flag, plus one additional flag).
2. Set CRC 16-bit CRC-CCITT.
3. Set DRT.
4. Set all other bits to zero or default.

For the PSMR definition, see Section 30.8, “HDLC Mode Register (PSMR).”

33.4.3 Programming the TODR

Use the transmit-on-demand (TODR) register to expedite a transmit frame. See Section 28.2.3, “Transmit-on-Demand Register (TODR).”

Chapter 34
QUICC Multi-Channel Controller (QMC)

The QUICC multi-channel controller (QMC) functionality can emulate up to 64 time-division serial channels, using a serial communication controller (SCC), and a time-division-multiplexed (TDM) physical interface. Each of the QMC channels can be independently programmed to perform in either HDLC or transparent mode.

Any available TDM in the serial interface (SI) can be used for the QMC protocol. The SI transfers the data between the TDM interface and the SCC. The SCC then performs the multiplexing/demultiplexing of the QMC channels. Figure 34-1 provides an overview of the QMC functionality.

Each SCC can work in QMC mode, either alone or together in any combination, spreading any of the 64 available QMC channels across the multiple SCCs. One TDM connection can be routed to one or more SCCs operating in QMC mode, with each SCC operating on different time slots.

It is also possible to use multiple TDMs for QMC with combined routing to one SCC or to separate SCCs. When using multiple TDMs connected to the same SCC, restrictions such as using common clocks and sync inputs apply; it is also important to avoid collisions by separating the serial interface (SI) routing, making sure that only one TDM will be accessing the SCC at any given time.

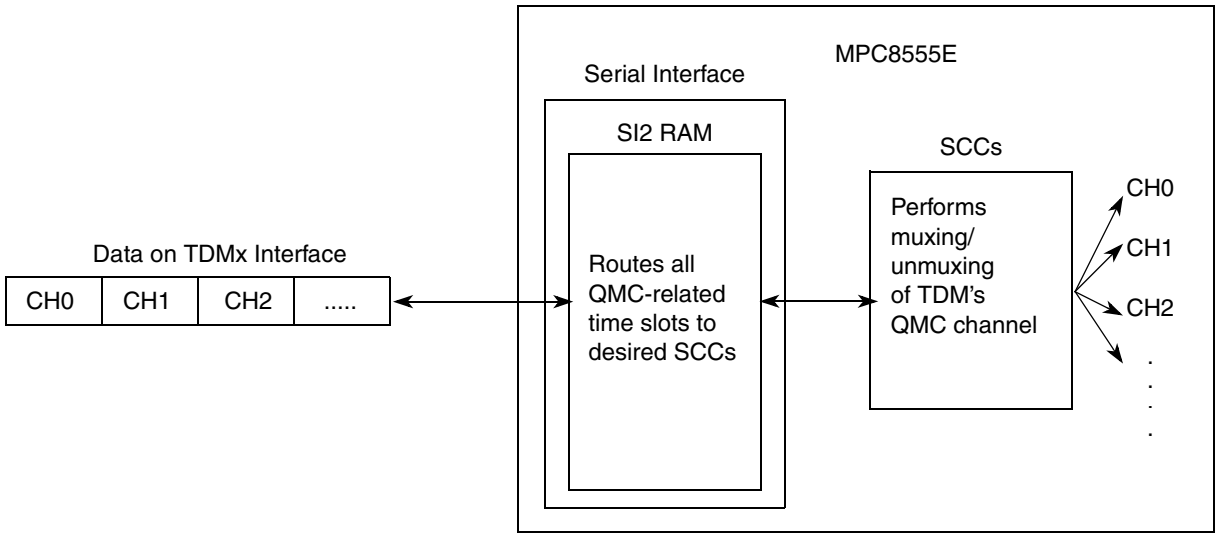


Figure 34-1. QMC Channel Addressing Capability

34.1 Features

QMC-specific features include the following:

- Up to 64 independent communication channels on one SCC, or split across multiple SCCs (64 channels total per device)
- Arbitrary mapping of any of 0–63 channels to any TDM time slot
- Can support up to two simultaneous 32-channel E1 links
- Independent mapping possible for receive/transmit
- Supports either transparent or HDLC protocols for each channel
- Interrupt circular buffer with programmable size and overflow identification
- Global loop mode
- Individual channel loop mode through the SI
- Programmable frame length through the SI

QMC features related to the serial interface include the following:

- Serial-multiplexed (full duplex) input/output 2048-, 1544-, or 1536-Kbps PCM highways
- Compatible with T1/DS1 24-channel and CEPT E1 32-channel PCM highway, ISDN basic rate, ISDN primary rate and user-defined
- Subchanneling on each time slot
- Allows independent transmit and receive routing, frame syncs, and clocking
- Concatenation of any, not necessarily consecutive, time slots to channels independently for receive/transmit
- Supports H0, H11, and H12 ISDN channels
- Allows dynamic allocation of channels

QMC features related to the system interface include the following:

- On-chip bus arbitration for serial DMAs with no performance penalty
- Efficient bus usage (no bus usage for nonactive channels and active channels that have nothing to transmit)
- Efficient control of the interrupts to the CPU
- Supports external buffer descriptors table
- Uses on-chip enlarged dual-ported RAM for parameter storage

34.2 QMC and the Serial Interface

QMC is designed to work in conjunction with the serial interface, taking advantage of its programmable SIRAM and additional functionalities. See Chapter 23, “Serial Interface with Time-Slot Assigner,” for details on proper programming of the SI’s registers and SIRAM. However, it is possible to operate QMC in nonmultiplexed serial interface (NMSI) mode, directly using the SCC’s own pins instead of the TDM interlace pins. Functions such as frame synchronization, loopback, echo, and inverted signals are performed in the serial interface and cannot be achieved in NMSI mode. It is recommended to use the serial interface even if only one SCC is used for the TDM bus.

Connecting an SCC to the SI or to its own pins in NMSI mode is selected by programming the CMX SCC clock route register (CMXSCR). See Section 24.4.4, "CMX SCC Clock Route Register (CMXSCR)," for details on proper programming of the CMXSCR.

This section describes additional functionality that the SI provides to QMC operation.

34.2.1 Synchronization

Independent receive and transmit clocks and frame synchronization signals control the data transfer. In NMSI operation, synchronization occurs only once after activating QMC, to initiate transfer using the CD (receive) and CTS (transmit) signals in pulse mode. If any noise corrupts either signal or the clock, the QMC will be out of synchronization until the whole protocol is restarted.

In contrast, the more robust SI performs a synchronization on each frame, limiting the damage from noise error on the clock or synchronization lines. Noisy channels can be restarted individually without interrupting other channels.

34.2.2 Loopback Mode

The loopback from a transmitter to a receiver can be implemented on a per QMC channel basis. If channel-specific loopback is desired, it is important to have each individual QMC time slot represented as an entry in the SIRAM in order to achieve proper operation. A common transmit and receive clock as well as a common frame synchronization pulse must be provided for loopback mode to work. The loopback is done on a fixed time slot of the actual TDM.

34.2.3 Echo Mode

The SI can be programmed to echo incoming data. In this mode, the complete TDM link is retransmitted from the incoming L1RXDx to the L1TXDx pin on a bit-by-bit basis. The receiver section of the selected SCC can operate normally and also receive the incoming bit stream. This is also known as global echo mode on the whole link. Individual time slot echo is not possible with QMC without software intervention.

34.2.4 Inverted Signals

All QMC-related receive and transmit data can be logically inverted by setting the RINV and TINV bits of the GSMR_L register. A logical inversion on a per channel basis is not possible in the QMC without external hardware. To invert a specific channel, the SI can be programmed to send a strobe signal at the QMC channel's corresponding time slot on the TDM interface. This strobe can then be connected to an external XOR gate to perform the inversion.

34.2.5 QMC Routing Changes On-The-Fly

Changes can be made on-the-fly in the QMC routing tables, but changes made to SI RAM require the QMC link to be disabled or require usage of a shadow RAM routing table. The shadow table can hold alternative routing information to be switched in at the appropriate time-slot boundary.

34.3 QMC Memory Organization

34.3.1 QMC Memory Structure

Figure 34-2 shows how data is addressed by the QMC protocol. It discusses addressing the dual-ported RAM to access data within the buffers.

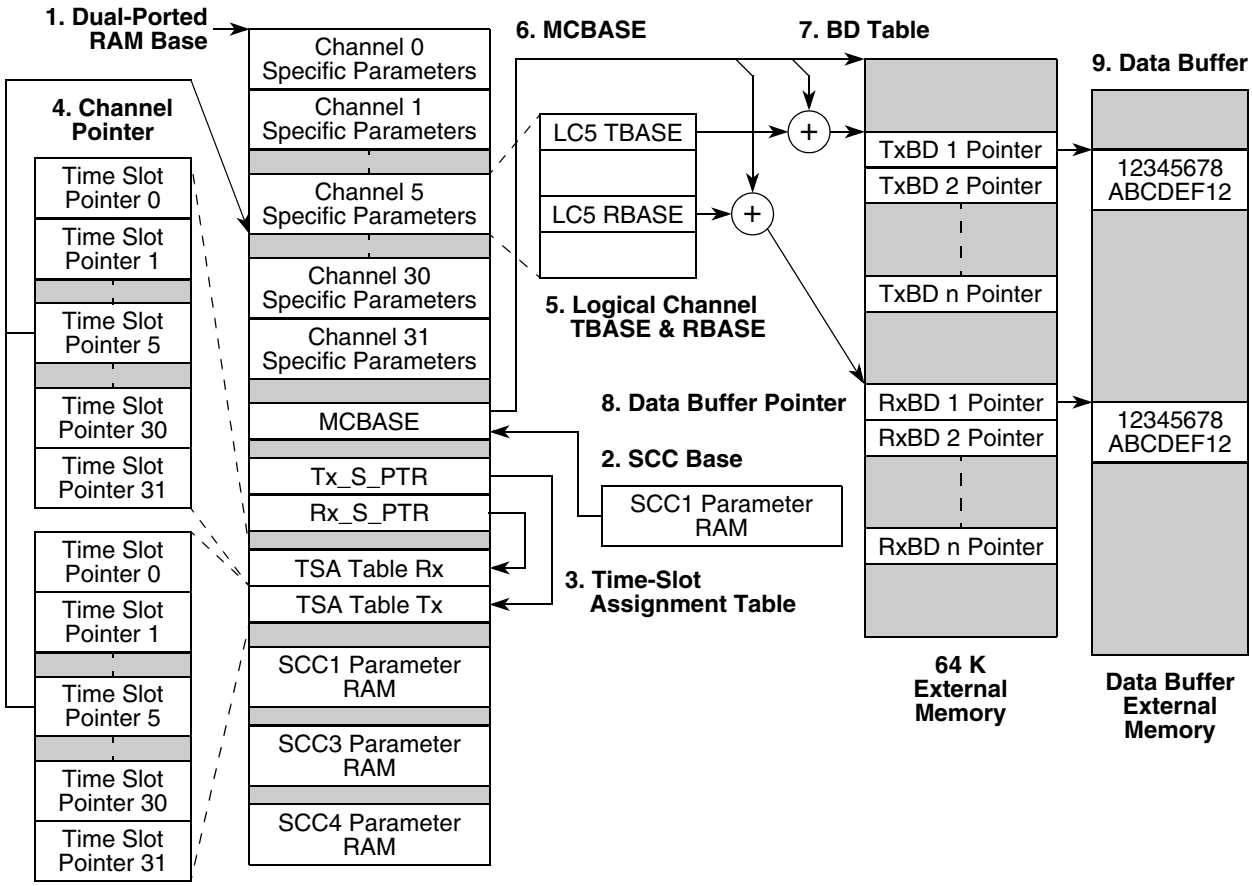


Figure 34-2. QMC Memory Structure

34.3.2 SCC Base and Global Multi-Channel Parameters

The SCC base points to the start of the parameter RAM for each of the SCCs at 256-byte intervals. When the QMC protocol is enabled on an SCC, its parameter RAM is used to store the global multi-channel parameters for all the logical channels. This area contains parameters and pointers that are common to all channels.

NOTE

As the QMC requires 0xAF bytes of parameter RAM for its global multi-channel parameters, this may cause conflict with other CPM functionality.

34.3.2.1 TSATRx/TSATTx Pointers and Time-Slot Assignment Table

The time-slot assignment table pointers are within the global multi-channel parameters. There are two pointers—Tx_S_PTR for transmit and Rx_S_PTR for receive. The Rx_S_PTR is normally set to SCC Base + 20; this is the normal location of the receive time-slot assignment table. The Tx_S_PTR is normally set to SCC Base + 60; this is the normal location of the transmit time-slot assignment table. However, if the receiver and the transmitter have the same mapping for the logical channels, Tx_S_PTR can point to SCC base + 20 so that Rx and Tx have a common time-slot assignment table. Note that if a single TDM channel is routed to more than one SCC, they may also use just one time-slot assignment table for all SCCs. See Section 34.3.3, "Multiple SCC Assignment Tables," for more information. The time-slot assignment table holds one 32-bit entry for each time-slot. It has options for subchanneling, a valid bit, and a logical channel pointer. For 64-channel support there is only space for one table; therefore, common Rx and Tx parameters will need to be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another SCC. Associated with the Rx/Tx_S_PTR are the Rx/TxPTR pointers that are maintained by the CPM and point to the current time slot.

34.3.2.2 TSATRx/TSATTx Channel Pointers

The channel pointers are 12-bit pointers to the channel-specific parameters in the internal dual-ported RAM. These should not be confused with TSATRx/TSATTx pointers as described in Section 34.3.2.1, "TSATRx/TSATTx Pointers and Time-Slot Assignment Table." The 6 most-significant bits of the address are taken from the time-slot assignment table. The 6 least-significant bits are zero, mapping out a 64-byte area for each of the channel-specific parameters. The channel-specific parameters are common for Rx and Tx. For 32-channel support, 2 Kbytes of dual-ported RAM is required (32 × 64), and for 64-channel support, 4 Kbytes of dual-ported RAM is required (64 × 64). In most cases, time slot 0 channel pointer will address the base of dual-ported RAM for logical channel 0, and time slot 1 channel pointer would address the base of dual-ported RAM + 4 for logical channel 1. In Figure 34-2, time slot 5 channel pointer addresses logical channel 5, requiring the channel pointer being set to 0b000101.

NOTE

It is possible to concatenate multiple time slots to one logical channel. This is achieved by setting the channel pointers of the grouped time slots to the same logical channel.

34.3.2.3 Logical Channel TBASE and RBASE

TBASE and RBASE are within the channel-specific parameters. TBASE is the Tx buffer descriptor base address, and RBASE is the Rx buffer descriptor base address. These 16-bit offsets from MCBASE point to individual logical channel's buffer descriptors located within the buffer descriptor table. Note that there are individual TBASE and RBASE values for each logical channel.

34.3.2.4 MCBASE

MCBASE is located in the global multi-channel parameters. Each SCC has a unique MCBASE value pointing to the base of the SCC's buffer descriptor table in external memory. For example, the address of logical channel five's Tx buffer descriptor table is MCBASE + logical channel five TBASE. MCBASE

normally points to external RAM, but it is permissible to set it up so that some or all BDs are placed within free areas of the DPRAM. This may save valuable access time if external memory is slow.

34.3.2.5 Buffer Descriptor Table

A buffer descriptor table for each SCC is located in a 64-Kbyte area of external memory. This block size is determined by the TBASE and RBASE addressing range. The memory segment must be long-word-aligned but can start anywhere in memory. Each SCC has a maximum of 16,384 (64 Kbytes memory ÷ 4-byte pointers) buffers. For a 32-channel implementation, each logical channel has a maximum of 256 (16,384 / (32 × 2)) buffers for receive and 256 buffers for transmit. For each logical channel, there is a circular queue with programmable start address and length.

34.3.2.6 Data Buffer Pointer

As with the standard CPM protocols, the data buffer is addressed by a 32-bit pointer within the buffer descriptor. This addresses the data received or transmitted from external memory.

34.3.2.7 Data Buffer

The data buffers in external memory can hold up to 64 Kbytes of data as determined by the data length in the buffer descriptor.

34.3.2.8 Global Multi-Channel Parameters

The global multi-channel parameters reside in the SCC's parameter RAM page and are common to all logical channels.

The largest portion of the global area is the time-slot assigner tables for the receiver and transmitter section of the SCC. For 32-channel support, there is one table for Tx and one for Rx within the parameter RAM. If the connection is split over multiple SCCs, this table only needs to be present once for multiple SCCs operating in QMC mode. See Section 34.3.3, "Multiple SCC Assignment Tables," for more information. For 64-channel support there is only space for one table; therefore common Rx and Tx parameters will need to be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another SCC.

The dual-ported RAM is used for the channel-specific area for all SCCs. It is important that individual time slots are mapped to only one SCC, and that individual logical channels are separated to avoid contention.

Table 34-1 lists the global parameters. Note that the boldfaced parameters must be initialized by the user. See Section 34.7, "QMC Initialization," for more information.

Table 34-1. Global Multi-Channel Parameters

Offset to SCC Base	Name	Width (Bits)	Description
00	**MCBASE**	32	Multi-channel base pointer—This host-initialized parameter points to the starting address of the 64-Kbyte buffer descriptor table in external memory. The MCBASE is used with the TBASE and RBASE registers in the channel-specific parameters.
04	**QMCSTATE**	16	Multi-channel controller state (initialize to 0x8000)—Internal QMC state machine value used by RISC processor for global state definition.
06	**MRBLR**	16	Maximum receive buffer length—This host-initialized entry defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. This parameter is only valid in HDLC mode. The buffer area allocated in memory for each buffer is MRBLR + 4. The QMC adds another long word if non-octet-aligned frames are received in HDLC operation. The non-octet information is written only to the last buffer of a frame, but it can happen in any buffer. See Section 34.6.1, “Receive Buffer Descriptor,” for more information. As the QMC works on long-word alignment, MRBLR value should be a multiple of 4 bytes.
08	**Tx_S_PTR**	16	Tx time-slot assignment table pointer (SCC base + 60 in normal mode; SCC base + 20 for common Rx & Tx time-slot assignment tables)—This global QMC parameter defines the start value of the TSATTx table. The TSATTx table in the global multi-channel parameter listing starts by default at SCC base + 60. Tx_S_PTR lets the user move the starting address of this table. If the same routing and masking are used for the transmitter and receiver, the tables can be overlaid, so Tx_S_PTR can point to SCC base + 20. This parameter is an offset from DPRBASE. This table must be present only once per SCC global area. Other SCCs can access this location.
0A	**RxPTR**	16	Rx pointer (initialize to SCC base + 20)—This global QMC parameter is a RISC variable that points to the current receiver time slot. The host must initialize this pointer to the starting location of TSATRx. The RISC processor increments this pointer whenever it completes the processing of a received time slot.
0C	**GRFTHR**	16	Global receive frame threshold—Used to reduce interrupt overhead when many short HDLC frames arrive, each causing an RXF interrupt. GRFTHR can be set to limit the frequency of interrupts. Set to 1 to get an interrupt per frame received. Note that the RXF event is written to the interrupt table on each received frame, but GINT is set only when the number of RXF events (by all channels) reaches the GRFTHR value. GRFTHR can be changed on-the-fly. For information about exception handling, see Section 34.5, “QMC Exceptions.”
0E	**GRFCNT**	16	Global receive frame count (initialized GRFCNT = GRFTHR)—A down-counter used to implement the GRFTHR feature. GRFCNT decrements for each frame received. No other receiver interrupts affect this counter. The counter value is set to the threshold during initialization. GRFCNT is automatically reset to the GRFTHR value by the CPM after a global interrupt.
10	**INTBASE**	32	Multi-channel interrupt base address (host-initialized)—This pointer contains the starting address of the interrupt circular queue in external memory. Each entry contains information about an interrupt request that has been generated by the QMC to the host. Each SCC operating in QMC mode has its own interrupt table in external memory. See Section 34.5, “QMC Exceptions.”
14	**INTPTR**	32	Multi-channel interrupt pointer (host-initialized)—This global parameter holds the address of the next QMC interrupt entry in the circular interrupt table. The RISC processor writes the next interrupt information to this entry when an exception occurs. The host must copy the value of INTBASE to INTPTR before enabling interrupts.

Table 34-1. Global Multi-Channel Parameters (continued)

Offset to SCC Base	Name	Width (Bits)	Description
18	**Rx_S_PTR**	16	Rx time-slot assignment table pointer (default = SCC base + 20 in normal mode)—This global QMC parameter defines the start value of the TSATRx table, which must be present only once per SCC global area. Other SCCs may access this location.
1A	**TxPTR**	16	TxPTR (initialize to SCC Base + 60)—This global parameter is a RISC variable that points to the current transmitter time slot. The host must initialize it to the starting location of TSATTx. The RISC processor increments this pointer whenever it completes the processing of a transmitter time slot.
1C	**C_MASK32**	32	CRC constant (0xDEBB20E3)—Required to calculate 32-bit CRC-CCITT. C_MASK32 is written by the host during QMC initialization. It is used for 32-bit CRC-CCITT calculation if HDLC mode of operation is chosen for a selected channel. (This is a programmable option. For each HDLC channel, one of two CRCs can be chosen, as programmed in CHAMR.) For more information, see Section 34.3.4.1, “Channel-Specific HDLC Parameters.” This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
20	**TSATRx**	32 entries x 16	Time slot assignment table Rx—Host-initialized, 16-bit-wide table with 32 entries that define mapping of logical channels to time slots for the QMC receiver. The QMC protocol looks at chunks of 8 bits regardless of whether they come from one physical time slot of the TDM or whatever other combination of bits the TSA transfers to the SCC. These 8 bits are referred to as a time slot in the assignment table. It is recommended but not required to route all bits from the TDM to the SCC and to do all enabling and masking in the time-slot assignment table. See Figure 34-3.
60	**TSATTx**	32 entries x 16	Time slot assignment table Tx—Maps a specific logical channel to each physical time slot. Time slot assignment table Tx is a host-initialized, 16-bit table with 32 entries that define the mapping of channels to time slots for the QMC transmitter. The QMC protocol looks at chunks of 8 bits regardless if they go to one physical time slot of the TDM or whatever other combination of bits are transferred from the SCC to the TDM through the TSA. These 8 bits are referred to as a time slot in the assignment table. It is recommended but not required to route all bits from the TDM to the SCC and to do all enabling and masking in the time-slot assignment table. See Figure 34-3.
A0	**C_MASK16**	16	CRC constant (0xF0B8)—Required to calculate 16-bit CRC-CCITT. This constant is written by the host during QMC initialization. It is used for 16-bit CRC-CCITT calculation if HDLC mode of operation is chosen for a selected channel. (This is a programmable option. For each HDLC channel, one of two CRCs can be chosen, as programmed in CHAMR.) For more information, see Section 34.3.4.1, “Channel-Specific HDLC Parameters.” This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
A4	TEMP_RBA	32	Temporary receive buffer address
A8	TEMP_CRC	32	Temporary cyclic redundancy check

Table 34-1. Global Multi-Channel Parameters (continued)

Offset to SCC Base	Name	Width (Bits)	Description
AC	RX_FRM_Base	16	This entry contains a pointer to an area in the DPR where the receiver Framer temporary parameters reside. The area designated for these parameters is a single byte per TDM channel. When the QMC is handling up to 32 channels it is possible to program this entry to the value SCC Base+0xC0 thus locating this data structure inside the SCC parameter page and save DPR space When using 64 channels this entry should point to a free area of 64 bytes in the DPR which is 64 bytes aligned.
AE	TX_FRM_Base	16	This entry contains a pointer to an area in the DPR where the receiver Framer temporary parameters reside. The area designated for these parameters is a single byte per TDM channel. When the QMC is handling up to 32 channels it is possible to program this entry to the value SCC Base+0xE0 thus locating this data structure inside the SCC parameter page and save DPR space When using 64 channels this entry should point to a free area of 64 bytes in the DPR which is 64 bytes aligned.

NOTE

There is no way the receiver and transmitter can share the same structure as done on the TSA table when the TDM functionality of the receiver is identical to this of the transmitter.

An area of 64 bytes starting at address SCC Base+0xC0 up to SCC Base+0xFF is available for this data structure.

The user must program the area pointed by RX_FRM_Base / TX_FRM_Base as follow:

Each 1-byte entry corresponds to a TDM channel numbered per the QMC time-slot assignment table. The number of entries is determined by the number of active channels in the system. Each time a channel is initialized, the corresponding entry should be programmed to this value.

NOTE

The area between SCC base + 20 and SCC base + 9F is normally used for TSA tables. The mapping above is ideal for 32-channel support. The exact mapping of the TSA tables is determined by the programming of Rx_S_PTR and Tx_S_PTR, and is not fixed. For 64-channel support it is suggested to use common Rx and Tx parameters. The TSA table will be common and have 64 entries starting at SCC base + 20; see Figure 34-4. Alternatively, another SCC's parameter RAM may be used, as determined by Rx_S_PTR and Tx_S_PTR; see Figure 34-6 for more information. However implemented, the TSA tables may reside anywhere in internal memory.

Figure 34-3 shows a general time-slot assignment table for 32 16-bit time slots. The fields will be used to either transmit or receive channels.

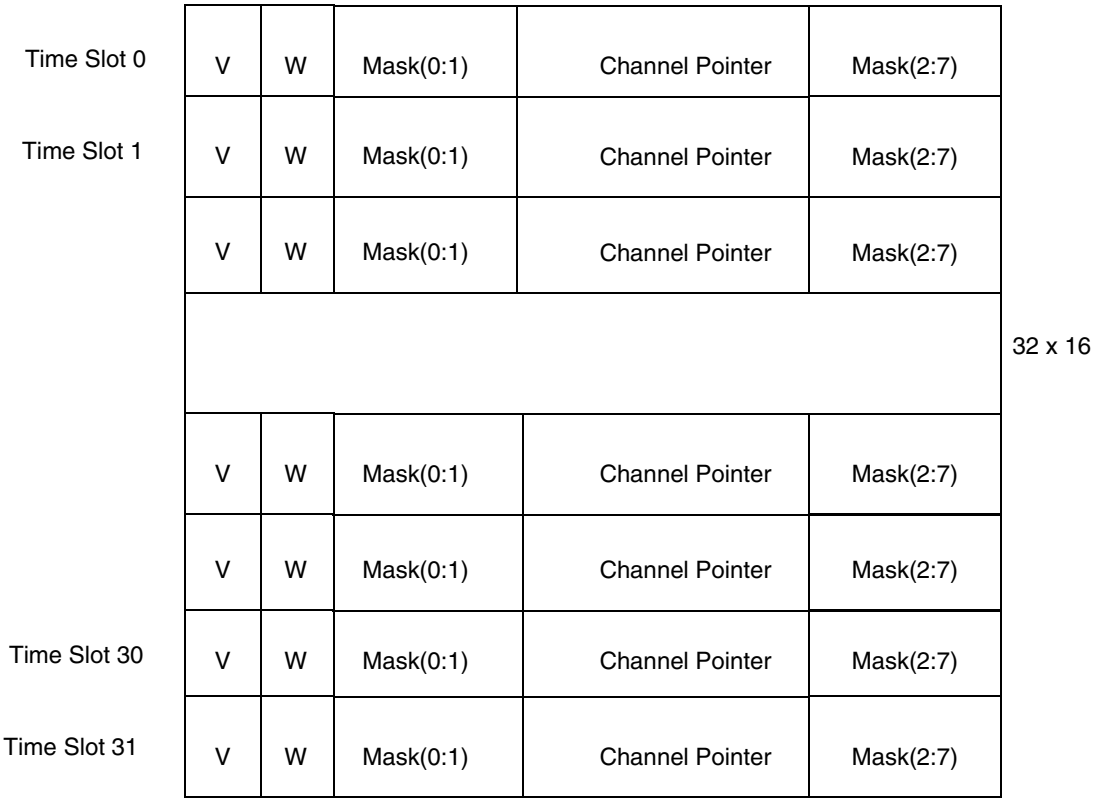


Figure 34-3. Time-Slot Assignment Table

Table 34-2 describes the fields in the time-slot assignment table for receive.

Table 34-2. Time-Slot Assignment Table Entry Fields for Receive Section

Field	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. 0 The data in this 8-bit time slot is totally ignored and not written to any buffer. 1 The data in this 8-bit time slot is valid and written to the current buffer, pointed to by the channel pointer entry, after protocol processing (for example, zero deletion in HDLC). Individual bits can be masked out as described later.
W	Wrap bit—Identifies the last entry in TSATRx 0 This is not the last time slot in the frame. 1 The RISC processor wraps around and handles time slot 0 or the first 8 bits transferred from the TSA in the next request. The next request is identified by a frame synchronization pulse.
Rx channel pointer	This 6-bit field of the TSATRx entry identifies the data channel routed to this time slot. The actual channel pointer is 12 bits long, and contains the starting address of the channel-specific parameter area (address of RBASE). The 6 most-significant bits are taken from the TSATRx channel pointer field, and the 6 least-significant bits are always internally set to zero.
Mask(0–7)	Mask bits—These 8 bits identify the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. Any unmasked bit (1) is processed in the receiver for a valid time slot. Any masked bit (0) is ignored by the receiver for a valid channel and no bit counter is affected.

Table 34-3 describes the fields in the time-slot assignment table for transmit.

Table 34-3. Time-Slot Assignment Table Entry Fields for Transmit Section

Name	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. 0 Logic 1 is transmitted. If the Tx signal of the TDM interface is programmed to be an open drain output (port B programming), other devices can transmit on nonvalid time slots. 1 Data is transmitted from its associated buffer in combination with the mask bit settings.
W	Wrap bit—The wrap bit identifies the last entry in TSATTx. 0 This is not the last time slot in the frame. 1 The RISC processor wraps around and handles time slot 0 or the first 8 bits of data in the SCC in the next request. The next request is identified by a frame synchronization pulse.
Tx channel pointer	This 6-bit field of the TSATTx entry identifies the data channel routed to this time slot. The actual channel pointer is 12 bits long, and contains the starting address of the channel-specific parameter area (address of TBASE). The 6 most-significant bits are taken from the TSATTx channel pointer field, and the 6 least-significant bits are always internally set to zero.
Mask(0–7)	Mask bits—Identifies the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. For a valid channel with an unmasked bit (1), the bit position is filled according to the protocol. A valid channel with a masked bit (0) transmits a logic high (1).

If the transmitter and receiver have the same mapping, it is possible to use a common time-slot assignment table. This is initialized by setting both Tx_S_PTR and Rx_S_PTR to SCC Base + 20. For 64-channel support it is suggested to use common Rx and Tx parameters. The time-slot assignment table will then also be common and have 64 entries starting at SCC Base + 20; see Figure 34-4.

Time Slot 0	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 1	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
					64 x 16
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 62	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 63	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)

Figure 34-4. Time-Slot Assignment Table for 64-Channel Common Rx and Tx Mapping

34.3.3 Multiple SCC Assignment Tables

Assume a scenario as depicted in Figure 34-5. A 2.048-Mbps TDM link is fed directly into the TSA. Within the SI RAM, the even channels (byte-wide) are muxed to SCC3 and the odd channels are muxed to SCC1. This arrangement is used to spread the load over two SCCs. This effectively doubles the FIFO depth on the QMC protocol. Time slots are switched to alternate SCCs to avoid data bursts that may stress the FIFOs. Each SCC sees a continuous bitstream without any gaps as described earlier.

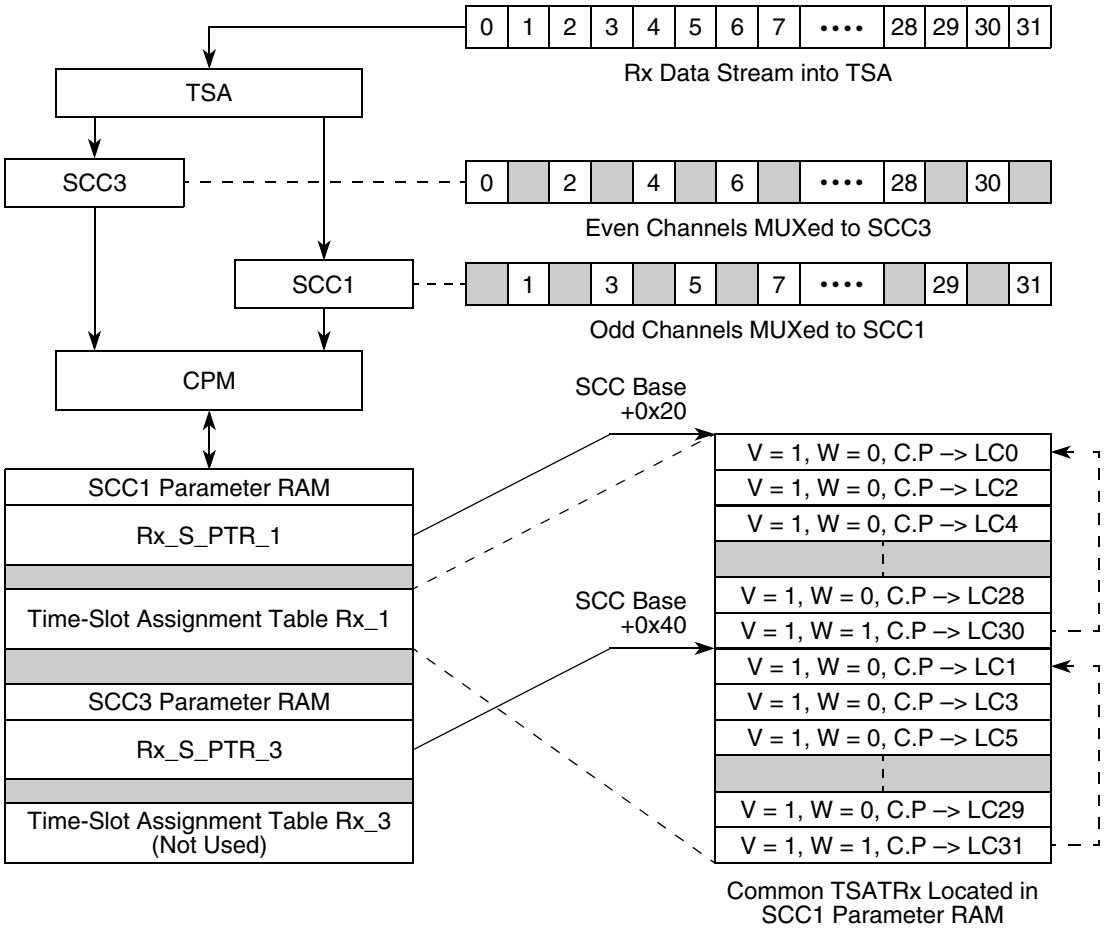


Figure 34-5. Rx Time-Slot Assignment Table for 32 Channels Over 2 SCCs

NOTE

It is important that multiples of bytes are routed to each SCC to delineate between time slots. Unused bits shall be routed to the SCC and be masked in the time-slot assignment table.

In Figure 34-5, each SCC has its own pointer, Rx_S_PTR_1 and Rx_S_PTR_3, addressing SCC1's time-slot assignment table. This table only needs to be present once in one of the SCC1's global parameter area. Rx_S_PTR_1 points to the start of the table, address SCC base + 20. The 16 logical channels from SCC1 are located in the first 16 entries of the table. The entry for logical channel 30 has the wrap bit (W) set, causing the CPM to wrap back to logical channel 0 on reception of the next byte routed to SCC1. Rx_S_PTR_3 addresses SCC base + 40, the start of the 16 entries for SCC3. The entry for logical channel 31 has the wrap bit (W) set, causing the CPM to wrap back to logical channel 1 on reception of the next byte routed to SCC3. Each entry within the table has a channel pointer to a logical channel. It is important that different SCCs do not point to the same logical channel.

The TSATTx is also located in SCC1's parameter RAM. This means that the area reserved for the TSA tables in SCC3's parameter RAM is free for alternative use.

A second scenario is depicted in Figure 34-6. A 4.096-Mbps TDM link is fed directly into the TSA. Again, within the SI RAM, the even channels (byte-wide) are muxed to SCC3 and the odd channels are muxed to SCC1. This arrangement is used to spread the load over two SCCs. Another reason this method may be used is to facilitate separate routing for the Rx and Tx logical channels. This requires two 64-entry tables that require 256 bytes, but only 128 bytes are allocated in the parameter RAM of an SCC for time-slot assignment tables. In this case, the Rx table is located in SCC1's parameter RAM, and the TX table is located in SCC3's parameter RAM, making most efficient use of memory.

Changes on-the-fly are easily accomplished by setting or clearing the valid bit for each time slot. Changes to the mask bits can also be made on-the-fly. This does not cause any problems to the QMC microcode itself, but may cause protocol errors on the channel in question depending on when this change is done.

It is possible to have a time-slot assignment table for every SCC in its corresponding RAM page and have all of the TDM routed to the different SCCs. This gives the user a very flexible system that can be changed on-the-fly without disconnecting the TDM interface. In this case the user must ensure that no collisions occur on the transmit lines from several SCCs.

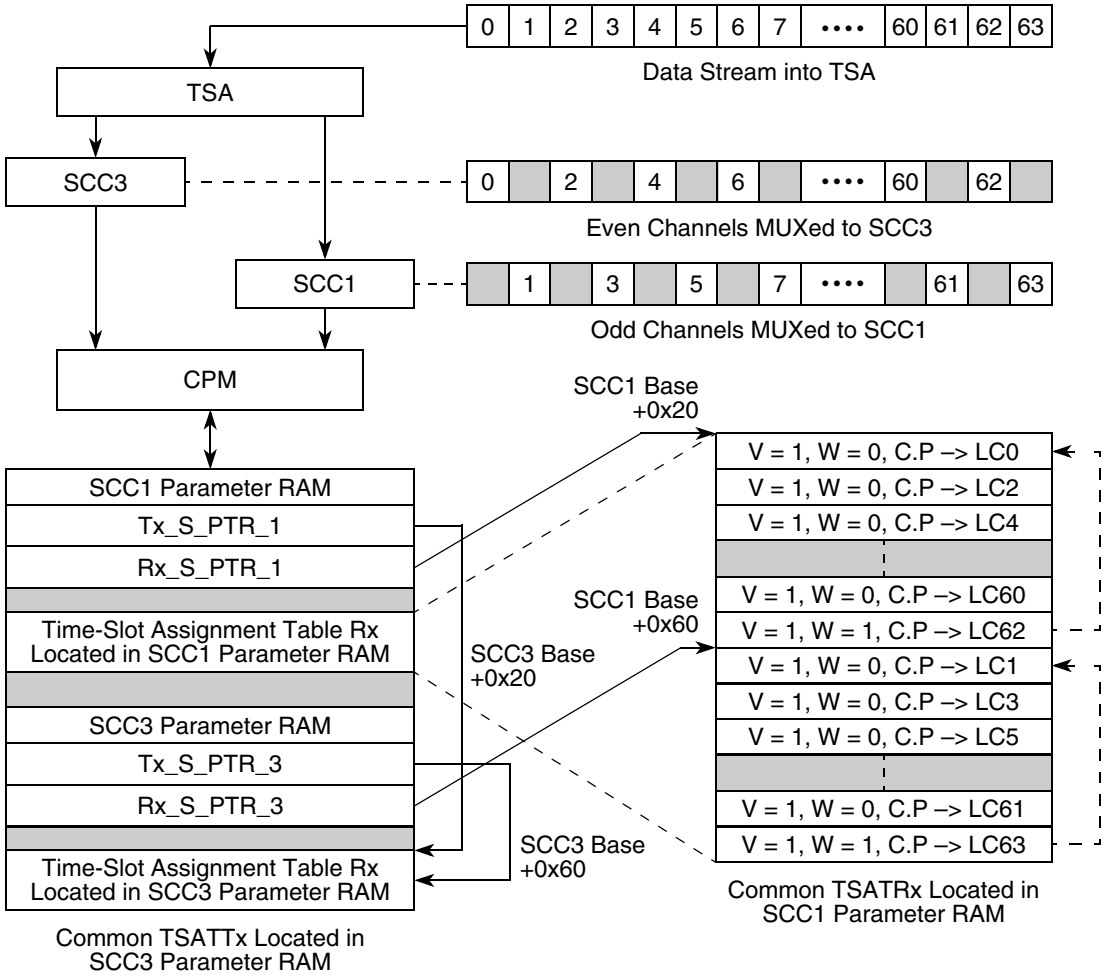


Figure 34-6. Time-Slot Assignment Tables for 64 Channels Over 2 SCCs

34.3.4 Channel-Specific Parameters

The channel-specific parameters are located in the lower part of the dual-ported RAM. Each channel occupies 64 bytes of parameters. Physical time slots can be matched to logical channels in several combinations. Unused logical channels leave a hole in the channel-specific parameters that can be used for buffer descriptors for the other SCCs.

The channel-specific area determines the operating mode—HDLC or transparent. Several entries take on different meanings depending on the protocol chosen.

34.3.4.1 Channel-Specific HDLC Parameters

Table 34-4 describes the channel-specific HDLC parameters. Boldfaced parameters must be initialized by the user.

Table 34-4. Channel-Specific HDLC Parameters

Offset	Name	Width (Bits)	Description
00	**TBASE**	16	Tx buffer descriptor base address—Offset of the channel's transmit buffer descriptor table relative to MCBASE, host-initialized. See Figure 34-2.
02	**CHAMR**	16	Channel mode register. See Section 34.3.4.1.1, "CHAMR—Channel Mode Register (HDLC)."
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Initialize before enabling the channel. See Section 34.3.4.1.2, "TSTATE—Tx Internal State (HDLC)."
08	—	32	Tx internal data pointer—Points to current absolute address of channel.
0C	**TBPTR**	16	Tx buffer descriptor pointer (host-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel again)—Offset of current BD relative to MCBASE. See Table 34-1. MCBASE + TBPTR gives the address for the BD in use.
0E	—	16	Tx internal byte count—Number of remaining bytes
10	TUPACK	32	(Tx Temp) Unpack 4 bytes from 1 long word
14	**ZISTATE**	32	Zero-insertion state (host-initialized to 0x0000_0200 for HDLC or transparent operation)—Contains the previous state of the zero-insertion state machine.
18	TCRC	32	Temp transmit CRC—Temp value of CRC calculation result
1C	**INTMSK**	16	Channel's interrupt mask flags—See Section 34.3.4.1.3, "INTMSK—Interrupt Mask (HDLC)."
1E	BDFlags	16	Temp
20	**RBASE**	16	Rx buffer descriptor offset (host-initialized)— Defines the offset of the channel's receive BD table relative to MCBASE (64-Kbyte table). See Figure 34-2.

Table 34-4. Channel-Specific HDLC Parameters (continued)

Offset	Name	Width (Bits)	Description
22	**MFLR**	16	Maximum frame length register (host-initialized)—Defines the longest expectable frame for this channel. Its maximum value is 64 Kbytes. The remainder of a frame which is larger than MFLR is discarded and a flag in the last frame's BD is set (LG). An interrupt request (RXF and RXB) might be generated depending on the interrupt mask. The frame length is considered to be everything between flags, including CRC. MFLR is checked every long word, but the content may be on any number of bytes. If MFLR is set to 5 for example, checking is done when 8 bytes have been received. At this point, the SDMA transfers the long word to memory, and all 8 bytes will be in the receive buffer. Also at this point the MFLR violation (>5) is detected and the interrupt may be generated. No more data will be written into this buffer when the MFLR violation is detected.
24	**RSTATE**	32	Rx internal state. See Section 34.3.4.1.4, "RSTATE—Rx Internal State (HDLC)," for more information.
28	—	32	Rx internal data pointer—Points to current address of specific channel.
2C	**RBPTR**	16	Rx buffer descriptor pointer (host-initialized to RBASE prior to operation or due to a fatal error)—Contains the offset from MCBASE to the current receive buffer. See Table 34-1. MCBASE + RBPTR gives the address for the BD in use.
2E	—	16	Rx internal byte count—Per Channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx Temp) Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0xFFFF_FFFF.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x80FF_FFE0 in HDLC mode prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of zero deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, hold the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel. More information is given in the application note section.
38	RCRC	32	Temp receive CRC—Temp value of CRC calculation result
3C	MAX_cnt	16	Max_length counter—Count length remaining
3E	TMP_MB	16	Temp—Holds MIN(MAX_cnt, Rx internal byte count)

34.3.4.1.1 CHAMR—Channel Mode Register (HDLC)

The channel mode register is a word-length, host-initialized register. Figure 34-7 shows the channel mode register for HDLC operation.

	0	1	2	3	4 6	7	8	9	10 11	12 15
Field	MODE	0	IDLM	ENT	Reserved	POL	CRC	0	Reserved	NOF
Reset	0000_0000_0000_0000									

Figure 34-7. CHAMR—Channel Mode Register (HDLC)

Table 34-5 describes the channel mode register's fields for HDLC operation. Boldfaced parameters must be initialized by the user.

Table 34-5. CHAMR Field Descriptions (HDLC)

Bits	Name	Description
0	**MODE**	Mode—Each channel has a programmable option whether to use transparent mode or HDLC mode. 0 Transparent mode 1 HDLC mode
1	—	0
2	**IDLM**	Idle mode. 0 Idle mode is disabled. No idle patterns are transmitted between frames. After transmitting the NOF + 1 flags, the transmitter starts the data of the frame. If between frames and the frame buffer is not ready, the transmitter sends flags until it can start transmitting the data. The NOF shall be greater or equal to the PAD setting; see Section 34.6.2, "Transmit Buffer Descriptor." If NOF = 0, this is identical to flag sharing in HDLC mode. For a high CPM load or with long bus latencies, the QMC protocol may insert additional flags. 1 Idle mode enabled. At least one idle pattern is transmitted between adjacent frames. If between frames and the frame buffer is not ready, the transmitter sends idle characters. When data is ready, the NOF + 1 flags are sent followed by the data frame. If in IDLE mode and NOF = 1, the following sequence is transmitted: .....init value, FF, FF, flag, flag, data,..... The init value before the idle will be 1's, in this case it is assumed the transmitter was uninitialized. An uninitialized SCC transmits 1s in every position.
3	**ENT**	Enable transmit. 0 Disable transmitter. If this bit is cleared and the channel's transmitter is routed to a certain time slot (within TSATTx, see Figure 34-3) the transmitter sends 1's on this time slot. 1 The transmit portion of the channel is enabled and data is sent according to protocol and to other control settings. Note that there is no ENR bit in the QMC protocol. To enable the receiver, the ZDSTATE and RSTATE parameters shall be set to their initial values.
4–6	—	Reserved
7	**POL**	Enable polling. This bit enables the transmitter to poll the transmit buffer descriptors. 0 The CPM does not check the ready bit (R) in the transmit buffer descriptor. 1 The CPM checks the ready bit (R) in the transmit buffer descriptor. The user can use this bit to prevent unnecessary external bus cycles when checking the ready bit (R) in the buffer descriptor. This bit should always be set by the software at the beginning of a transmit sequence of one or more frames. This bit is cleared (0) by the RISC processor when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared (0), that is, at the end of a frame or at the end of a multi-frame transmission. In order to prevent deadlock the software should always prepare the new BD, or multiple BDs, and set (1) the ready bit in the BD, before setting (1) the POL bit. Note that as this bit is automatically cleared by the CPM; the user should not attempt to clear this bit in software.
8	**CRC**	This bit selects the type of CRC when using the HDLC channel mode. 0 16-bit CCITT-CRC is selected for this channel 1 32-bit CCITT-CRC is selected
9	—	0
10–11	—	Reserved
12–15	**NOF**	Number of flags—Defines the minimum number of flags before frames. However, even if NOF = 0, at least one flag is transmitted before the first frame. See the description of the IDLM bit for more information.

34.3.4.1.2 TSTATE—Tx Internal State (HDLC)

TSTATE defines the internal transmitter state. The high byte of TSTATE defines the function code/address type. Figure 34-8 shows the TSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		TC2	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 34-8. TSTATE—Tx Internal State (HDLC)

Table 34-6 describes TSTATE fields.

Table 34-6. TSTATE Field Descriptions (HDLC)

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	TC2	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared.

34.3.4.1.3 INTMSK—Interrupt Mask (HDLC)

Each event defined in the interrupt circular queue entry maps directly to a bit in INTMSK as shown in Figure 34-10. There is one mask bit for each event—NID (bit 2), IDL (bit 3), MRF (bit 10), UN (bit 11), RXF (bit 12), BSY (bit 13), TXB (bit 14) and RXB (bit 15). Bits that do not map to an event are reserved. Reserved bits must be set to zero. Refer to Section 34.5, “QMC Exceptions,” for more detail.

- 0 = No interrupt request is generated and no new entry is written in the circular interrupt table.
- 1 = Interrupts are enabled.

This register is initialized by the host prior to operation.

	0	1	2	3	4	9	10	11	12	13	14	15
Field	V	W	NID	IDL	Channel Number		MRF	UN	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000											

Figure 34-9. Interrupt Table Entry

	0	1	2	3	4	9	10	15
Field	Reserved		Interrupt Mask		Reserved		Interrupt Mask Bits	
Reset	0000_0000_0000_0000							

Figure 34-10. INTMSK (HDLC)

34.3.4.1.4 RSTATE—Rx Internal State (HDLC)

The RSTATE is host-initialized before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command.

The high byte of RSTATE defines the function code/address type. Figure 34-11 shows the RSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		TC2	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 34-11. RSTATE—Rx Internal State (HDLC)

Table 34-7 describes RSTATE fields.

Table 34-7. RSTATE Field Descriptions (HDLC)

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	TC2	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared.

34.3.4.2 Channel-Specific Transparent Parameters

Table 34-8 describes the channel-specific transparent parameters. Boldfaced parameters must be initialized by the user.

Table 34-8. Channel-Specific Transparent Parameters

Offset	Name	Width	Description
00	**TBASE**	16	Tx buffer descriptor base address—Defines the offset of the channel's transmit BD table relative to MCBASE, host-initialized. See Figure 34-2.
02	**CHAMR**	16	Channel mode register. See Section 34.3.4.2.1, "CHAMR—Channel Mode Register (Transparent Mode)."
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Initialize before enabling the channel. See Section 34.3.4.2.2, "TSTATE—Tx Internal State (Transparent Mode)."
08		32	Tx internal data pointer—Points to current absolute address of channel.
0C	**TBPTR**	16	Tx buffer descriptor pointer (host-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel)—Contains the offset of current BD relative to MCBASE. See Table 34-1. MCBASE + TBPTR gives the address for the BD in use.
0E		16	Tx internal byte count—Number of remaining bytes
10	TUPACK	32	(Tx temp) Unpack 4 bytes from 1 long word
14	**ZISTATE**	32	Zero-insertion machine state (host-initialized to 0x0000_0200)—Contains the previous state of the zero-insertion state machine.
18	RES	32	—
1C	**INTMSK**	16	Channel's interrupt mask flags. See Figure 34-15.
1E	BDFlags	16	Temp
20	**RBASE**	16	Receive buffer descriptor base offset—Defines the offset of the channel's 64-Kbyte receive BD table relative to MCBASE. Host-initialized. See also Figure 34-2.
22	**TMRBLR**	16	Transparent maximum receive buffer length (host-initialized entry)—Defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. Note that this value must be a multiple of 4 bytes as the QMC works on long-word alignment.
24	**RSTATE**	32	Rx internal state. See Section 34.3.4.2.5, "RSTATE—Rx Internal State (Transparent Mode)," for more information.
28		32	Rx internal data pointer—Points to current address of specific channel.
2C	**RBPTR**	16	Rx buffer descriptor pointer (host-initialized to RBASE, prior to operation or due to a fatal error)—Contains the offset from MCBASE to the current receive buffer. See Figure 34-2. MCBASE + RBPTR gives the address for the BD in use.
2E		16	Rx internal byte count—per channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx temp)—Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0xFFFF_FFFF.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x02FF_33E0 in transparent mode prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of the zero-deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, holds the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel.
38	RES	32	—

Table 34-8. Channel-Specific Transparent Parameters (continued)

Offset	Name	Width	Description
3C	**TRNSYNC**	16	Transparent synchronization—In transparent mode, this register controls synchronization for single time slots or superchannel applications. See Section 34.3.4.2.4, "TRNSYNC—Transparent Synchronization (Transparent Mode)."
3E	RES	16	—

34.3.4.2.1 CHAMR—Channel Mode Register (Transparent Mode)

The channel mode register is a word-length, host-initialized register. Figure 34-12 shows the channel mode register for transparent mode.

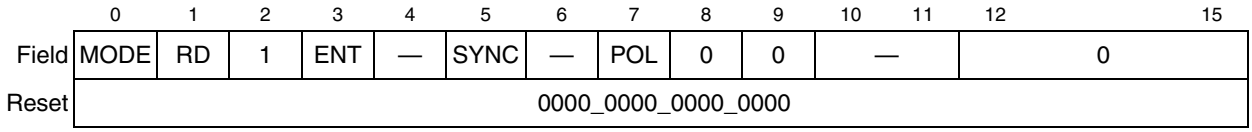


Figure 34-12. CHAMR—Channel Mode Register (Transparent Mode)

Table 34-9 describes the channel mode register's fields for transparent operation. Boldfaced parameters must be initialized by the user.

Table 34-9. CHAMR Bit Settings (Transparent Mode)

Bits	Name	Description
0	**MODE**	Mode—Each channel has a programmable option whether to use transparent mode or HDLC mode. 0 Transparent mode 1 HDLC mode
1	**RD**	Reverse data 0 The bit order will not be reversed, transmitting/receiving the LSB of each octet first. 1 The bit order as seen on the channels is reversed, transmitting/receiving the MSB of each octet first.
2	—	1
3	**ENT**	Enable transmit 0 Disable transmitter. If this bit is cleared and the channel's transmitter is routed to a certain time slot (within TSATTx, see Figure 34-3.) the transmitter sends 1's on this time slot. 1 The transmit portion of the channel is enabled and data is sent according to protocol and to other control settings.
4	—	Reserved
5	**SYNC**	Synchronization—Controls synchronization of multi-channel operation in transparent mode. 0 The first byte is put in the first available time slot or is read from the first available time slot to this logical channel. 1 The synchronization algorithm according to TRANSYNC is done.
6	RES	Reserved

Table 34-9. CHAMR Bit Settings (Transparent Mode) (continued)

Bits	Name	Description
7	**POL**	Enable polling—Enables the transmitter to poll the transmit BDs. 0 The CPM will not check the ready (R) bit in the transmit buffer descriptor. 1 The CPM will go check the ready (R) bit in the transmit buffer descriptor. The user can use this bit to prevent unnecessary external bus cycles when checking the ready bit (R) in the buffer descriptor. Software should always set POL at the beginning of a transmit sequence of one or more frames. The RISC processor clears POL (0) when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared (0), that is, at the end of a frame or at the end of a multi-frame transmission. To prevent deadlock, software should prepare the new BD, or multiple BDs, and set (1) the ready (R) bit in the BD before setting (1) POL. Note that the CPM automatically clears this bit; the user should never try to clear this bit in software.
8–9	—	0
10–11	—	Reserved
12–15	—	0

34.3.4.2.2 TSTATE—Tx Internal State (Transparent Mode)

TSTATE defines the internal transmitter state. The high byte of TSTATE defines the function code/address type. Figure 34-13 shows the TSTATE register for transparent mode.

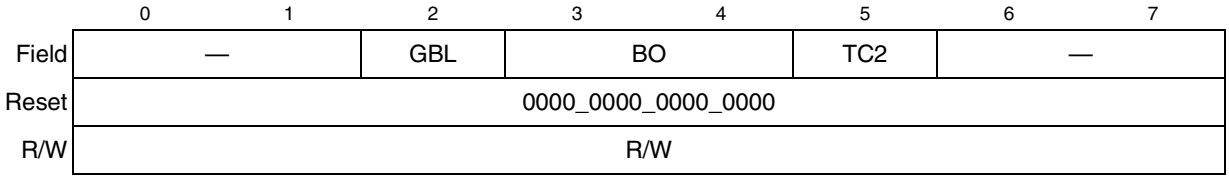


Figure 34-13. TSTATE—Tx Internal State (Transparent Mode)

Table 34-10 describes TSTATE fields.

Table 34-10. TSTATE Field Descriptions (Transparent Mode)

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	TC2	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared.

34.3.4.2.3 INTMSK—Interrupt Mask (Transparent Mode)

Each event defined in the interrupt circular queue entry maps directly to a bit in INTMSK as shown in Figure 34-15. There is one mask bit for each event—UN (bit 11), BSY (bit 13), TXB (bit 14), and RXB (bit 15). Bits that do not map to an event are reserved. Reserved bits must be set to zero.

- 0 = No interrupt request is generated and no new entry is written in the circular interrupt table.
- 1 = Interrupts are enabled.

This register is initialized by the host before operation.

	0	1	2–4	5–9	10	11	12	13	14	15
Field	V	W	—	Channel Number	—	UN	—	BSY	TXB	RXB
Reset	0000_0000_0000_0000									

Figure 34-14. Interrupt Table Entry

	0–10	11–15
Field	Reserved	Interrupt Mask Bits
Reset	0000_0000_0000_0000	

Figure 34-15. INTMSK (Transparent Mode)

34.3.4.2.4 TRNSYNC—Transparent Synchronization (Transparent Mode)

In transparent mode, the TRNSYNC register controls the synchronization for single time slots or superchannel applications.

NOTE

This register has no meaning if the SYNC bit in the channel mode register (CHAMR) is cleared (0).

When sending a transparent message over several time slots, it is necessary to know in which time slot the first byte of data appears.

The TRNSYNC word-length register is divided into two parts with the high byte controlling the first received time slot and the low byte controlling the transmitter synchronization.

Take the example of a superchannel of several time slots:

TS*n*, TS*n* + 1, TS*n* + 2 ..... TS*n* + x

The algorithm for the receiver byte in decimal is:

(TS*n* + 1) × 2

The algorithm for the transmit byte in decimal is:

(TS*n* + x + 1) × 2

The result from these calculations is a decimal value programmed into TRNSYNC.

NOTE

Note that TS*n* is not necessarily the first time slot in the frame. For example, if a superchannel is produced from TS2, TS4, and TS6, the message may be arranged with TS4 holding the first byte, then TS6, and the final byte held in TS2 of the following frame.

The following nine cases in Figure 34-16, named C1 to C9, show different scenarios ranging from a single time slot per logical channel to a superchannel using several time slots. In this application, 24 time slots are routed to this SCC from the SI RAM. After time slot 23, the frame starts with 0 again. The arrow in all the figures illustrates the starting position.

C1 is for a single byte in TS7, so TS*n* = 7

Rx Byte: $(7+1) \times 2 = 16$

As x = 0, TS*n* + x = TS*n* = 7, so

Tx Byte: $(7 + 1) \times 2 = 16$

C2 is a single byte in TS23, so TS*n* = 23. Note that time slot after 23 is 0, so in the calculations below 23 + 1 = 0.

Rx Byte: $(23 + 1) \times 2 = 0$

As x = 0, TS*n* + x = TS*n* = 23, so

Tx Byte: $(23 + 1) \times 2 = 0$

C3 is a 2-byte pattern TS7, TS8, so TS*n* = 7

Rx Byte: $(7 + 1) \times 2 = 16$

As x = 1, TS*n* + x = 8, so

Tx Byte: $(8 + 1) \times 2 = 18$

C4 is a 2-byte pattern TS8, TS7, so TS*n* = 8

Rx Byte: $(8 + 1) \times 2 = 16$

As x = 1, TS*n* + x = 7, so

Tx Byte: $(7 + 1) \times 2 = 16$

C5 is a 2-byte pattern TS19, TS23, so TS*n* = 19

Rx Byte: $(19 + 1) \times 2 = 40$

As x = 1, TS*n* + x = 23, so

Tx Byte: $(23 + 1) \times 2 = 0$

C6 is a 2-byte pattern TS23, TS19, so TS*n* = 23

Rx Byte: $(23 + 1) \times 2 = 0$

As x = 1, TS*n* + x = 19, so;

Tx Byte: $(19 + 1) \times 2 = 40$

C7 is a 4-byte pattern TS20, TS23, TS8, TS9, and TS19, so TS*n* = 20

Rx Byte: $(20 + 1) \times 2 = 42$

As x = 5, TS*n* + x = 19, so

Tx Byte: $(19 + 1) \times 2 = 40$

C8 is a 4-byte pattern TS8, TS9, TS19, TS20, and TS23, so TS*n* = 8

Rx Byte: $(8 + 1) \times 2 = 18$

As x = 5, TS*n* + x = 23, so

Tx Byte: $(23 + 1) \times 2 = 0$

C9 is a 4-byte pattern TS19, TS20, TS23, TS8, and TS9, so TS*n* = 19

Rx Byte: $(19 + 1) \times 2 = 40$

As x = 5, TS*n* + x = 9, so

Tx Byte: $(9 + 1) \times 2 = 20$

NOTE

Case C1 and C2 can be used as described above. A more elegant solution for single time slot applications is to have the SYNC bit cleared (0) in the channel mode register. Both scenarios produce the same result.

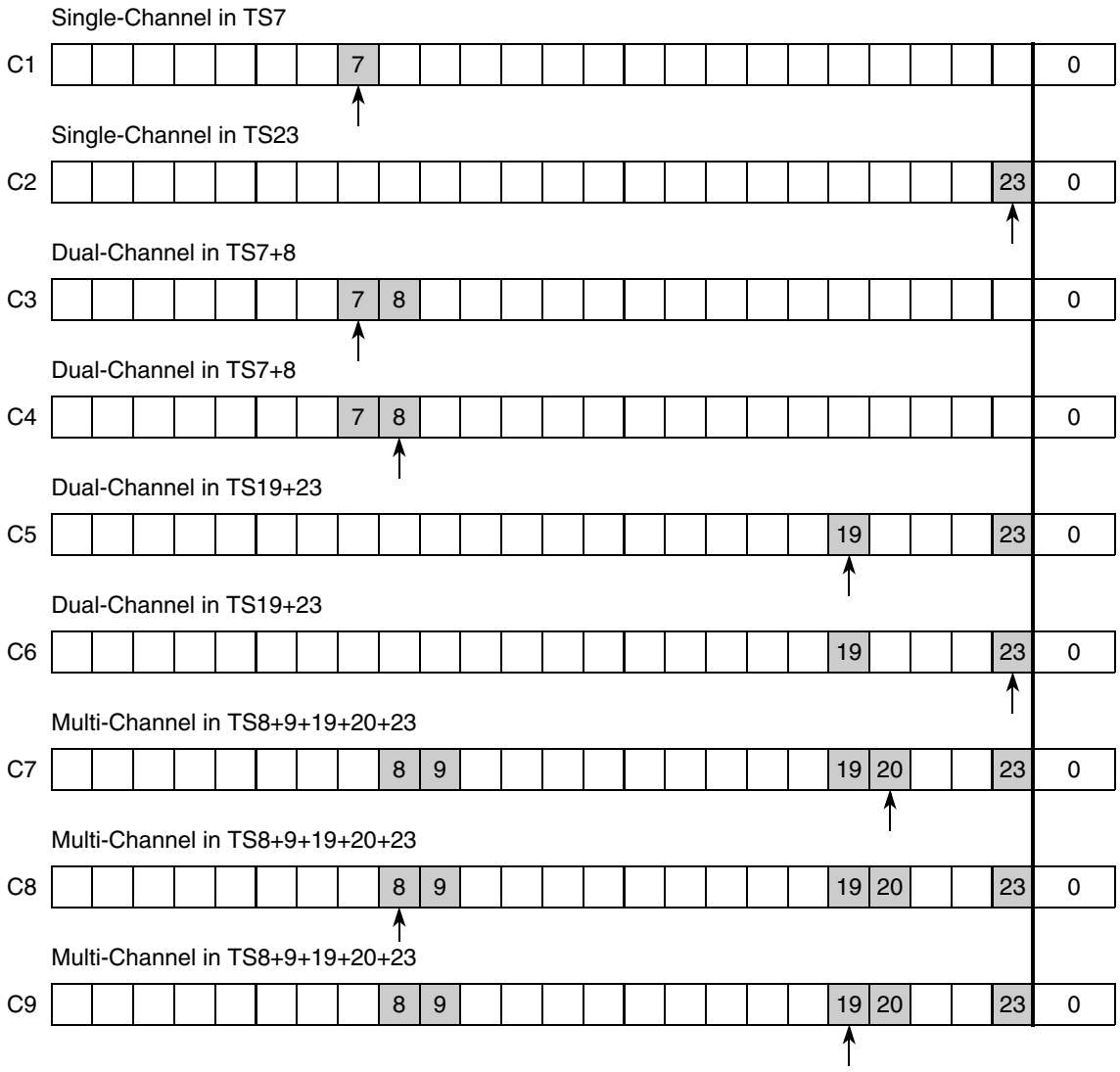


Figure 34-16. Examples of Different T1 Time-Slot Allocation

34.3.4.2.5 RSTATE—Rx Internal State (Transparent Mode)

The RSTATE is host-initialized before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command.

The high byte of RSTATE defines the function code/address type. Figure 34-17 shows the RSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		TC2	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 34-17. RSTATE—Rx Internal State (Transparent Mode)

Table 34-11 describes RSTATE fields.

Table 34-11. RSTATE Field Descriptions (Transparent Mode)

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	TC2	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared.

34.4 QMC Commands

The host issues commands to the QMC by writing to the command register (CPCR). Refer to Section 21.3.1, “CP Command Register (CPCR).”

34.4.1 Transmit Commands

STOP TRANSMIT <channel>

The stop transmit command disables the transmission of data on the selected channel and clears CHAMR[POL]. Upon asserting this command in the middle of a frame, the RISC processor sends an ABORT indication (7F) followed by IDLEs or FLAGs, depending on the mode, on the selected channel. If this command is issued between frames, the RISC processor continues sending IDLEs or FLAGs (depending on CHAMR[IDLM]) in this channel.

The Tx buffer descriptor pointer (TBPTR) is not advanced to the next buffer; see Table 34-4 and Section 34.3.2.8, “Global Multi-Channel Parameters.”

Set the CHAMR[POL] bit to 1 to start transmission or to continue after a stop command.

Only after transmission starts for a deactivated channel, which is identified by a cleared V bit in the time-slot assignment table or a cleared ENT bit, is it necessary to initialize ZISTATE and TSTATE before setting CHAMR[POL].

To deactivate a channel, clear both the V bit in the TSA table and CHAMR[ENT].

34.4.2 Receive Commands

STOP RECEIVE <channel>

The stop receive command forces the receiver of the selected channel to stop receiving. After issuing this command, the microcode does not change any of the receive parameters in the dual-ported RAM. The command immediately stops activity on the channel. It does not wait for a frame reception to be finished. Because the current buffer descriptor can remain open if a reception was in progress, it is possible to get errors on this buffer once you restart reception. When restarting, set ZDSTATE first and RSTATE afterwards.

Initialize ZDSTATE and RSTATE to their initial values to start reception or to continue receiving after a stop command.

NOTE

No commands exist to start transmission and reception. This function is realized through the GSMR register.

34.5 QMC Exceptions

QMC interrupt handling involves two principle data structures—the SCC event register (SCCE) and the circular interrupt table. Figure 34-18 illustrates the circular interrupt table.

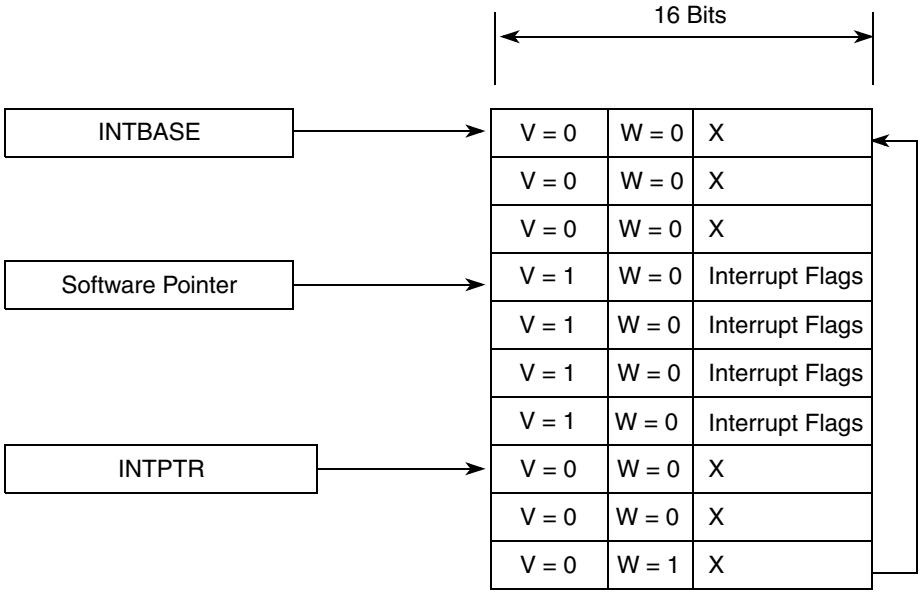


Figure 34-18. Circular Interrupt Table in External Memory

INTBASE (interrupt base) points to the starting location of the queue in external memory, and INTPTR (interrupt pointer) marks the current empty position available to the RISC processor. Both pointers are host-initialized global QMC parameters; see Table 34-1. The entry whose W (wrap) bit is set to 1 marks the end of the queue. When one of the QMC channels generates an interrupt request, the RISC processor writes a new entry to the queue. In addition to the channel's number, this entry contains a description of

the exception. The V (valid) bit is then set and INTPTR is incremented. When INTPTR reaches the entry with W = 1, INTPTR is reset to INTBASE.

An interrupt is written to the interrupt table only if it survives a mask with the INTMASK (interrupt mask) register. Following a write to the queue, the QMC protocol updates the SCC event register (SCCE) according to the type of exception.

Following a request that is not masked out by the INTMASK or the SCCM (SCC mask) register, an interrupt is generated to the host. The host reads the SCCE to determine the cause of interrupt. A dedicated SCCE bit (GINT) indicates that at least one new entry was added to the queue. After clearing GINT, the host starts processing the queue. The host then clears this entry's valid bit (V). The host follows this procedure until it reaches an entry with V = 0, indicating an invalid entry.

NOTE

It is very important that the user clear all bits but the wrap bit in a queue entry even though its valid bit may be cleared. Clearing only the interrupt bits confuse user software with incorrect channel interrupts.

34.5.1 Global Error Events

A global error affects the operation of the SCC. A global error can occur for two reasons—serial data rates being too high for the CPM to handle, and CPM bus latency being too long for correct FIFO operation.

There are two global errors—global transmitting underrun (GUN) and global receiver overrun (GOV). GUN indicates that transmission has failed due to lack of data; and GOV indicates that the receiver has failed because the RISC processor did not write previous data to the receive buffer. In both cases, it is unknown which channel(s) are affected.

Nonglobal, individual channel errors are handled differently. See Section 34.5.3, "Interrupt Table Entry," for underrun and overrun in a specific channel.

The incoming data to the CPM is governed by transfers between the SCC and the SI. Every transfer in either direction causes a request to the CPM state machine. If requests are received too quickly, the CPM may lose track of the mapping of serial data to the QMC channels. If this happens, the CPM has to cause a global error to halt activity on all QMC channels until user software intervenes. Note that this problem typically indicates a performance-related design problem. Also note that latency is very important.

The other error condition is bus latency. A receiving channel submits data to the FIFO for transfer to external memory as long as the channel operates normally. If the bus latency for the SDMA channels is too long and the receive FIFO is filled and overwritten, a receiver overflow occurs. The overwriting channels cannot be traced, affecting entire QMC operation.

A similar situation can occur during transmission when the SDMA cannot fill the FIFO from external memory because of bus latency. Again, it cannot be determined which channel is underrun, and the whole QMC operation is affected.

Global errors are unlikely to occur in normal system operation, if correct serial speed is used. The only area of concern is data movement between the FIFO and external memory. To avoid problems, the user must understand the bus arbitration mechanism of the QUICC and meet the latency requirements.

34.5.1.1 Global Underrun (GUN)

The QMC performs the following actions when it detects a GUN event:

- Transmits an abort sequence of minimum sixteen 1's in each time slot
- Generates an interrupt request to the host (if enabled) and sets the GUN bit in the SCCE register
- Stops reading data from buffer
- Sends IDLEs or FLAGs in all time slots depending on channel mode settings until the host does the following:

  Host initializes all transmitting channels and time slots by preparing all buffer descriptors for transmission (R bits are set) and setting the POL bit. No other re-initialization is needed.

34.5.1.2 Global Overrun (GOV) in the FIFO

A global overrun affects all channels operating from an SCC. Following GOV, the QMC performs the following:

- Updates the RSTATE register to prevent further reception on this channel. Bit 20 in the RSTATE register indicates that the receiver is stopped.
- Generates an interrupt request to the host (if enabled) and sets the GOV bit in the SCCE
- Stops writing data to all channel's buffers
- Waits for host to initialize all the receiving channels by setting first the ZDSTATE followed by the RSTATE to their initial values

34.5.1.3 Restart from a Global Error

The last two bullets in the above two sections describe the only steps necessary for re-initialization. The transmit and receive sections must be restarted individually for each separate logical channel.

For details about initialization, see Section 34.7, "QMC Initialization."

34.5.2 SCC Event Register (SCCE)

The QMC's SCCE is a word-length register used to report events and generate interrupt requests. See Figure 34-19 and Table 34-12 for SCCE field descriptions. For each of its flags, a corresponding programmable mask/enable bit in the SCCM determines whether an interrupt request is generated. If a bit in the SCCM register is zero, the corresponding interrupt flag does not survive, and the CPM does not proceed with its usual interrupt handling. If a bit in the SCCM is set, the corresponding interrupt flag in the SCCE survives, and the SCC event bit is set in the CPM interrupt-pending register. See Figure 34-20 for SCCM assignments.

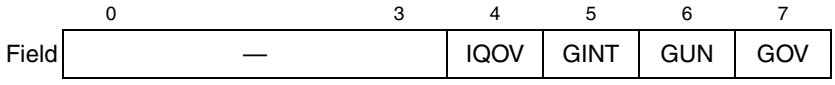


Figure 34-19. SCC Event Register

Table 34-12. SCC Event Register Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4	IQOV	Interrupt table (interrupt queue) overflow 0 No interrupt table overflow has occurred. 1 An overflow condition in the circular interrupt table occurs (and an interrupt request is generated). This condition occurs if the RISC processor attempts to write a new interrupt entry into an entry that was not handled by the host. Such an entry is identified by V = 1. This bit should be cleared immediately after reading the SCCE and recognizing this condition by writing a 1 to its location in the SCCE. If this condition occurs, the interrupt last received is lost. It does not overwrite the first entry that is still to be handled by the CPU.
5	GINT	Global interrupt 0 No global interrupt has occurred. 1 This flag indicates that at least one new entry in the circular interrupt table has been generated by the QMC. The host clears GINT by writing a 1 to its location in SCCE. After clearing it, the host reads the next entry from the circular interrupt table, and starts processing a specific channel's exception. The user must make sure that no more valid interrupts are pending in the interrupt table after clearing the GINT bit, before performing the RTE to avoid deadlock. This procedure ensures that no pending interrupts exist in the queue.
6	GUN	Global transmitter underrun 0 No global transmitter underrun has occurred. 1 This flag indicates that an underrun occurred in the SCC's transmitter FIFO. This error is fatal since it is unknown which channel(s) are affected. Following the assertion of the GUN bit in the SCCE, the QMC stops transmitting data on all channels. The TDM Tx line goes into idle mode. This error affects only the transmitter; the receiver continues to work. After initializing all the individual channels, the host may resume transmitting. If enabled in the SCCM, an interrupt request is generated when GUN is set. The host may clear GUN by writing 1 to its location in the SCCE.
7	GOV	Global receiver overrun 0 No global receiver overrun has occurred. 1 This flag indicates that an overrun occurred in the SCC's receiver FIFO. This error is fatal since it is unknown which channel(s) are affected. Following the assertion of the GOV bit in the SCCE, the QMC stops receiving data on all channels. Data is no longer written to memory. This error affects only the receiver; the transmitter continues to work. After initializing all the individual channels, the host may resume receiving. If enabled in SCCM, an interrupt request is generated when GOV is set. The host may clear GOV by writing 1 to its location in the SCCE.

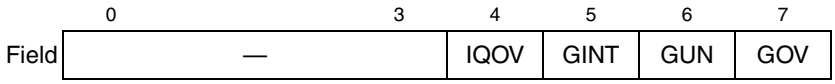


Figure 34-20. SCCM Register

34.5.3 Interrupt Table Entry

The interrupt table contains information about channel-specific events. Its flags are shown in Figure 34-21. Note that some bits have no meaning when operating in transparent mode. For more detailed description on which bits are used in HDLC and transparent operation, refer to Section 34.3.4, "Channel-Specific Parameters." Table 34-13 describes the fields of an interrupt table entry.

	0	1	2	3	4 – 9	10	11	12	13	14	15
Field	V	W	NID	IDL	Channel Number	MRF	UN	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000										

Figure 34-21. Interrupt Table Entry

Table 34-13. Interrupt Table Entry Field Descriptions

Bits	Name	Description
0	V	Valid bit 0 Entry is not valid. 1 Valid entry containing interrupt information. Upon generating a new entry, the RISC processor sets this bit. The V bit is cleared by the host immediately after it reads the interrupt flags in this entry (before processing the interrupt). The V bits in the queue are host-initialized. During the initialization procedure, the host must clear those bits in all queue entries.
1	W	Wrap bit 0 This is not the last entry in the circular interrupt table. 1 This is the last circular interrupt table entry. The next event's entry is written/read (by RISC/host) from the address contained in INTBASE. During initialization, the host must clear all W bits in the queue except the last one which must be set. The length of the queue is left to the user and can be a maximum of 64 Kbytes.
2	NID	Not idle 0 No NID event has occurred. 1 A pattern which is not an idle pattern was identified. NID interrupts are not generated in transparent mode.
3	IDL	Idle 0 No IDL event has occurred. 1 The channel's receiver has identified the first occurrence of HDLC idle (FFFE) after any non-idle pattern. IDL interrupts are not generated in transparent mode.
4–9	Channel Number	Identifies the requesting logical channel index (0–31).
10	MRF	Maximum receive frame length violation—This interrupt occurs in HDLC mode when more than MFLR number bytes are received. As soon as MFLR is exceeded, this interrupt is generated and the remainder of the frame is discarded. At this point the receive buffer is not closed and the reception process continues. The receive buffer is closed upon detecting a flag. The length field written to this buffer descriptor is the entire number of bytes received between the two flags. MRF interrupts are not generated in transparent mode. **Note**: The MRF interrupt is generated directly when the MFLR value is a multiple of 4 bytes. The checking of this is done on a long-word boundary whenever the SDMA transfers 32 bits to memory. If MFLR is not aligned to 4x bytes, this interrupt may be 1- to 3-byte timings late for this channel. In any case, the violation can be checked to any number of bytes. The last entry in the data buffer is always a full long word.

Table 34-13. Interrupt Table Entry Field Descriptions (continued)

Bits	Name	Description
11	UN	Tx no data 0 No UN event has occurred. 1 There is no valid data to send to the transmitter. The transmitter sends an abort indication consisting of 16 consecutive 1's and then sends idles or flags according to the protocol and the channel mode register setting. This error occurs when a transmit channel has no data buffer ready for a multi-buffer transmission already in progress. Transmission of a frame is a continuous bitstream without gaps or interruption. When a buffer is not ready in the middle of this sequence, it is an error situation. This can be viewed as channel underrun. The transmitter for this channel is stopped. See Section 34.7.1, "Restarting the Transmitter," for recovery information.
12	RXF	Rx frame 0 No RXF event has occurred. 1 A complete HDLC frame is received. Data is stored in external memory and the buffer descriptor is updated. If during frame reception an abort sequence of at least seven 1's is detected, the buffer is closed and both RXB and RXF are reported along with the AB in the buffer descriptor. As a result of end-of-frame, the global frame counter GRFCNT is decremented for interrupt generation. This counter is decremented on RXF only, regardless if the RXF was caused by correct closing or due to an error. RXF interrupts are not generated in transparent mode.
13	BSY	Busy 0 No BSY event has occurred. 1 A frame was received but was discarded due to lack of buffers. This can be viewed as channel overrun. After a busy condition, the receiver for this channel is disabled and no more data is transferred to memory. Receiver restart is described in Section 34.7.2, "Restarting the Receiver."
14	TXB	Tx buffer 0 No TXB event has occurred. 1 A buffer has been completely transmitted. This bit is set (and an interrupt request is generated) as soon as the programmed number of PAD characters (or the closing flag, for PAD = 0) is written to the SCC's transmit FIFO. The number of PAD characters determines the timing of the TXB interrupt in relation to the closing flag sent out at TXD. See Section 34.6, "Buffer Descriptors," for an explanation of PAD characters and their use.
15	RXB	Rx buffer 0 No RXB event has occurred. 1 A buffer has been received on this channel.

34.5.4 Interrupt Handling

To handle interrupts by the QMC, first read the SCCE register. Write back immediately all the bits that you recognize and handle. This clears the respective interrupt events. It is, for example, incorrect to first handle all the new interrupt table entries and clear GINT afterwards. To avoid deadlocks in your software, you must clear the recognized interrupt bits in the SCCE before actually handling the interrupt entries. Clear only the bits that you handle. Never clear bits for events in the SCCE that you do not handle. This facilitates debugging.

Upon receiving a new GINT event, always handle all the new entries in the queue, not just a single one. After handling an entry, make sure that the entry is completely cleared out with the exception of the Wrap bit. Any entry that does not have the Valid bit set must be completely cleared out, including the channel number. If all the entries are not cleared out on startup or after handling, these bits will confuse the software

when the entry is used again. QMC will only OR in new bits and numbers. It will not overwrite and clear bits that were set previously.

34.5.5 Channel Interrupt Processing Flow

Figure 34-22 illustrates the flow of a channel interrupt. Note that this does not describe the processing of the global interrupts GUN and GOV.

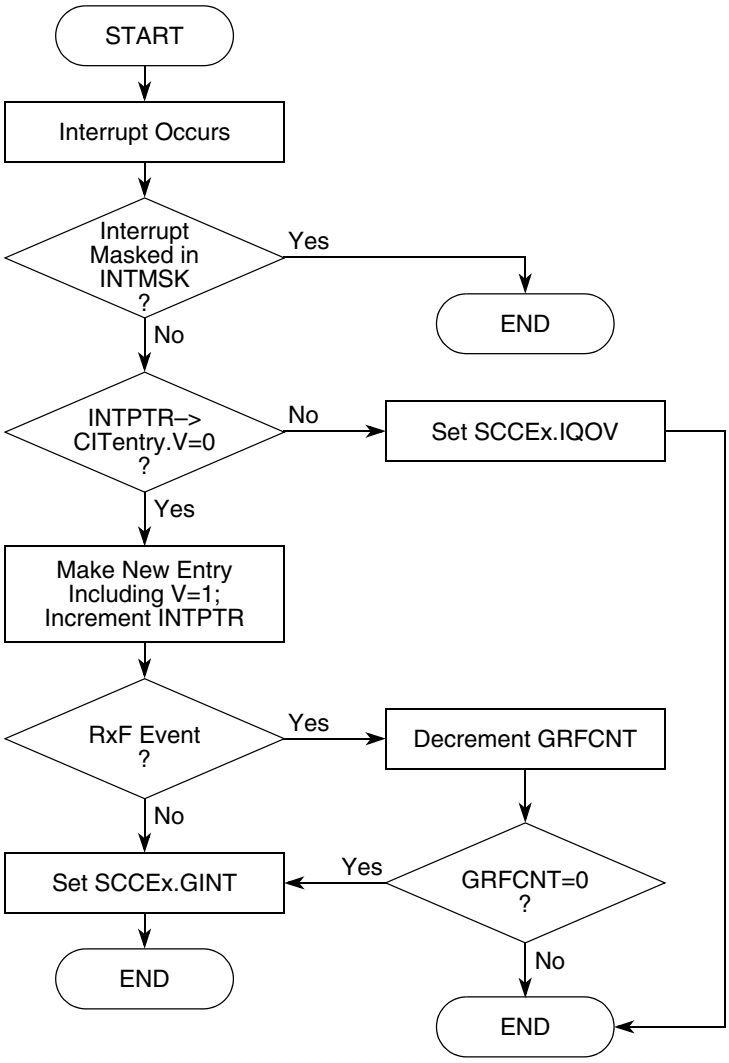


Figure 34-22. Channel Interrupt Flow

34.6 Buffer Descriptors

QMC buffer descriptors are located within 64 Kbytes in external memory; see Figure 34-2. Each buffer descriptor contains key information about the buffer it defines.

The first two sections describe the contents of the receive and transmit buffer descriptors for the QMC protocol.

The third section discusses the placement of QMC and non-QMC buffer descriptors in internal and external memory.

NOTE

The CPM 'ORs' the various status bits into the BD. You must clear all the status bits generated by the CPM before (re-)enabling the BD or you may confuse your own software with left over old status values.

34.6.1 Receive Buffer Descriptor

Figure 34-23 shows a receive buffer descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	E	—	W	I	L	F	CM	—	UB	—	LG	NO	AB	CR	—	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	Rx DATA BUFFER POINTER															
OFFSET + 6																

Note: Entries in boldface must be initialized by the user.

Figure 34-23. Receive Buffer Descriptor (RxBD)

Table 34-14 describes the individual fields of a receive buffer descriptor. Boldfaced entries must be initialized by the user.

Table 34-14. RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The user is free to examine or write to any fields of this RxBD. The CP does not use this BD again while the empty bit remains zero. 1 The data buffer associated with this BD is empty, or reception is in progress. This RxBD and its associated receive buffer are in use by the CP. When E = 1, the user should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table. After this buffer has been used, the CP receives incoming data into the first BD in the table (the BD pointed to by RBASE). The number of RxBDs in this table is programmable and is determined by the wrap bit.
3	**I**	Interrupt 0 The RXB bit is not set after this buffer has been used, but RXF operation remains unaffected. 1 The RXB or RXF bit in the HDLC interrupt circular table entry is set when this buffer has been used by the HDLC controller. These two bits may cause interrupts (if enabled).

Table 34-14. RxBD Field Descriptions (continued)

Bits	Name	Description
4	L	Last in frame (only for HDLC mode of operation). The HDLC controller sets L = 1, when this buffer is the last in a frame. This implies the reception either of a closing flag or of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 This buffer is not the last in a frame. 1 This buffer is the last in a frame.
5	F	First in frame. The HDLC controller sets F = 1 for the first buffer in a frame. In transparent mode, F indicates that there was a synchronization before receiving data in this BD. 0 This is not the first buffer in a frame. 1 This is the first buffer in a frame.
6	**CM**	Continuous mode 0 Normal operation (The empty bit (bit 0) is cleared by the CP after this BD is closed.) 1 The empty bit (bit 0) is not cleared by the CP after this BD is closed, allowing the associated data buffer to be overwritten automatically when the CP next accesses this BD. However, if an error occurs during reception, the empty bit is cleared regardless of the CM bit setting.
7	—	Reserved, should be cleared.
8	**UB**	User bit. UB is a user-defined bit that the CPM never sets nor clears. The user determines how this bit is used.
9	—	Reserved, should be cleared.
10	LG	Rx frame length violation (HDLC mode only). Indicates that a frame length greater than the maximum value was received in this channel. Only the maximum-allowed number of bytes, MFLR rounded to the nearest higher word alignment, are written to the data buffer. This event is recognized as soon as the MFLR value is exceeded when data is word-aligned. When data is not word-aligned, this interrupt occurs when the SDMA writes 64 bits to memory. The worst-case latency from MFLR violation until detected is 7 bytes timing for this channel. When MFLR violation is detected, the receiver is still receiving even though the data is discarded. The buffer is closed upon detecting a flag, and this is considered to be the closing flag for this buffer. At this point, LG is set (1) and an interrupt may be generated. The length field for this buffer is everything between the opening flag and this last identifying flag.
11	NO	Rx nonoctet-aligned frame. A frame of bits not divisible exactly by eight was received. NO = 1 for any type of nonalignment regardless of frame length. The shortest frame that can be detected is of type FLAG-BIT-FLAG, which causes the buffer to be closed with NO error indicated. The Non Octet byte is not written into memory and the data length DL field is not updated with this count.
12	AB	Rx abort sequence. A minimum of seven consecutive 1s was received during frame reception. Abort is not detected between frames. The sequence Closing-Flag, data, CRC, AB, data, opening-flag... does not cause an abort error. If the abort is long enough to be an idle, an idle line interrupt may be generated. An abort within the frame is not reported by a unique interrupt but rather with a RXF interrupt and the user has to examine the BD.
13	CR	Rx CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.
14	—	Reserved, should be cleared.
15	—	Reserved, should be cleared.

NOTE: RxBD and MRBLR

When the length of a received frame is an exact multiple of MRBLR, it is possible that the CP will not mark the BD that contains the last of the frame's data as the last BD in the frame. The CP will close the BD, mark it with E = 0, set the Data Length field to be the length of the data in this buffer, but leave L (bit 4) cleared. Then, the CP will close the next BD, erroneously mark it as containing data (E = 0), and mark it as the last BD in the frame (L = 1).

Figure 34-24 shows how non-octet alignment is reported and how data is stored. The two diagrams on the left show the reception of a single-buffer, 12-byte frame including the CRC. In the top case, the reception is correctly octet-aligned and the frame length indicates 12 bytes.

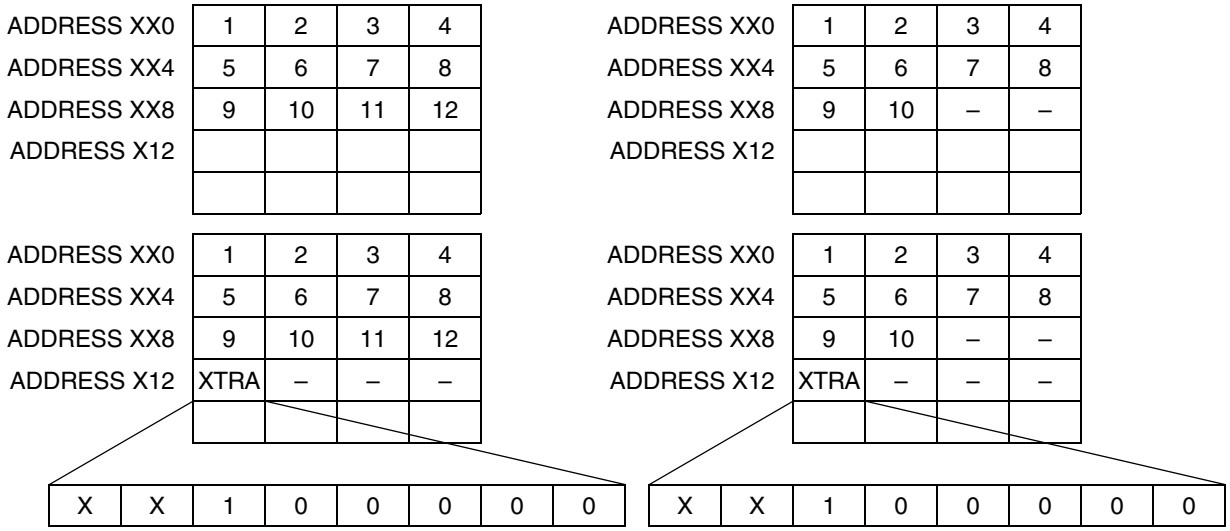


Figure 34-24. Nonoctet Alignment Data

In the bottom case, two more bits are received. The frame length is now 13 bytes, and the address positions X13 through X15 point to invalid data. Address position X12 contains information about the non-octet alignment. Valid information is written starting at the MSB position, shown as 'x' in the diagram. Starting from the LSB position, zeros are filled in followed by a '1' immediately preceding the valid data.

The two diagrams on the right show how the data and the extra information is stored for a frame length that is not a multiple of 4 bytes. The additional information is always on a long-word boundary. In the top case the frame length is 10 bytes and in the bottom case the length is 11 bytes.

For a minimum frame consisting of 'flag, 1 bit, flag' the frame length is 1. The only valid entry is at address XX0 with content of x1000000.

To accommodate the extra long word that may be written at the end of a frame and to avoid overwritten memory beyond the end of a buffer, it is recommended to reserve MRBLR + 8 bytes for each buffer area. The XTRA information shown in Figure 34-24 is written as 32-bit word. Under special, but quite possible circumstances the XTRA data is written four bytes further than indicated. Therefore, users must allocate MRBLR+8 bytes for each buffer area for QMC to avoid the risk of these memory areas being overwritten with XTRA info.

34.6.2 Transmit Buffer Descriptor

Figure 34-25 shows the transmit buffer descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	R	—	W	I	L	TC	CM	—	UB	—	—	—	PAD			
OFFSET + 2	DATA LENGTH															
OFFSET + 4	Tx DATA BUFFER POINTER															
OFFSET + 6																

Note: Entries in boldface must be initialized by the user.

Figure 34-25. Transmit Buffer Descriptor (TxBD)

Table 34-15 describes the individual fields of a transmit buffer descriptor. Boldfaced entries must be initialized by the user.

Table 34-15. Transmit Buffer Descriptor (TxBD) Field Descriptions

Bits	Name	Description
0	**R**	Ready 0 The data buffer associated with this buffer descriptor is not ready for transmission. The user can manipulate this buffer descriptor or its associated data buffer. The CPM clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is being transmitted. If R = 1, the user cannot write to fields of this buffer descriptor.
1	—	—
2	**W**	Wrap (final buffer descriptor in table) 0 This is not the last buffer descriptor in the TxBD table. 1 This is the last descriptor in the Tx buffer descriptor table. After this buffer is used, the CPM transmits data from the first buffer descriptor in the table (the buffer descriptor pointed to by TBASE). The number of TxBDs in this table is programmable and is determined only by the wrap bit and the overall space constraints of the dual-ported RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 TXB in the circular interrupt table entry is set when the controller services this buffer. This bit can cause an interrupt (if enabled).
4	**L**	Last 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
5	**TC**	Tx CRC (HDLC mode only). This bit is valid only when L = 1; otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used for testing purposes to send an erroneous CRC after the data. 1 Transmit the CRC sequence after the last data byte.
6	**CM**	Continuous mode 0 Normal operation 1 The R bit is not cleared by the CPM after this buffer descriptor is closed, allowing the associated data buffer to be retransmitted automatically when the CPM next accesses this buffer descriptor.
7	—	—

Table 34-15. Transmit Buffer Descriptor (TxBD) Field Descriptions (continued)

Bits	Name	Description
8	**UB**	User bit—The CPM never touches, sets, or clears this user-defined bit. The user determines how this bit is used. For example, it can be used to signal between higher-level protocols whether a buffer has been processed by the CPU.
9–11	—	—
12–15	**PAD**	Padding bits—These four bits indicate the number of PAD characters (0x7E or 0xFF depending on IDLM mode in the CHAMR register) that the transmitter sends after the closing flag. The transmitter issues a TXB interrupt only after sending the programmed value of pads to the Tx FIFO. The user can use the PAD value to guarantee that a TXB interrupt occurs after the closing flag has been sent on the TXD line. PAD = 0 means the TXB interrupt is issued immediately after sending the closing flag to the Tx FIFO. The number of PAD characters depends on the FIFO size and the number of time slots in use. An example explains the calculation: In SCC1 the FIFO is 32 bytes. If 16 time slots are used in the link, the resulting number of PAD characters is 32/16 = 2, to append to this buffer to ensure that the TXB interrupt is not given before the closing flag has been transmitted through the TXD line. The number of PAD characters must not exceed the NOF characters, ensuring that the closing of one buffer (the interrupt generation) occurs before the start of the next frame (clearing of R-bit). After the sequence of a closing flag followed by (PAD + 1) flag characters, a TXB interrupt will be generated; see Figure 34-26.
16–31	**DL**	Data length—The data length is the number of bytes the CPM should transmit from this buffer descriptor's data buffer. It is never modified by the CPM. This field should be greater than zero.
32–63	**TxBP**	Tx buffer pointer—The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory. This value is never modified by the CPM.

Figure 34-26 shows a TXB interrupt generated after (PAD + 1) flag characters following the closing flag. Four flags (NOF = 3) precede the next data. To set up this sequence correctly, the PAD value must not exceed NOF.

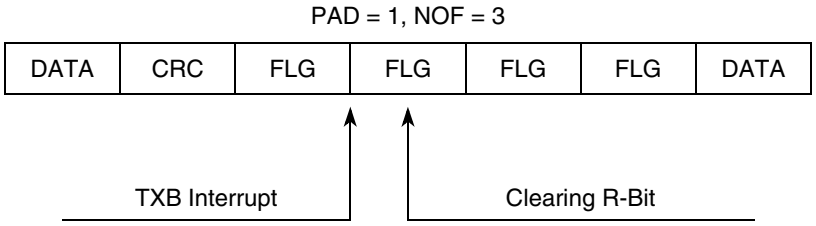


Figure 34-26. Relation Between PAD and NOF

34.6.3 Placement of Buffer Descriptors

The internal dual-ported RAM is used to store the buffer descriptors for all non-QMC operation. This solution causes minimum loading of the external bus. When starting any operation or switching between buffers during operations, several accesses must be made by the CPM to find the actual data buffers and to read and write control and status information. This process is unseen by the user for internal accesses, and any external bus master or memory refresh control can occur uninterrupted.

34.6.3.1 Parameter RAM Usage for QMC Over Several SCCs

There are two possible memory configurations for operating the QMC protocol on several SCCs. For each SCC in QMC protocol, a global parameter area is used, consuming at most 190 bytes if every global area contains the routing tables. The time-slot assignment tables (TSATRx and TSATTx) together occupy 128 bytes. The tables for each SCC in the parameter RAM can be duplicated, requiring 190 bytes per SCC.

Alternatively, multiple SCCs can use one set of common time-slot assignment tables (TSATRx and TSATTx), as described in Section 34.3.2.1, "TSATRx/TSATTx Pointers and Time-Slot Assignment Table." One SCC RAM page uses 190 bytes, 62 bytes for parameter fields and 128 bytes for the common time-slot assignment tables, allowing the other SCCs to use only 62 bytes of parameter RAM for QMC protocol, freeing their 128 bytes of time-slot assignment table space.

34.6.3.2 Internal Memory Structure

For details about the MPC8555E internal memory, refer to Chapter 2, "Memory Map," and Section 21.4, "Internal RAM."

To support 32 channels, only 2-Kbyte dual-ported RAM is needed for channel-specific parameters. Each logical channel occupies 64 bytes; thus 32 channels require 2 Kbytes, leaving 2 Kbytes free in the dual-ported RAM for buffer descriptors for other protocols.

To support 64 channels, 4-Kbyte dual-ported RAM is required for channel-specific parameters. Each logical channel occupies 64 bytes, requiring 4 Kbytes for 64 channels.

Non-QMC protocol implementations may be constrained by these memory requirements since their buffer descriptors need to use internal memory space.

If fewer than 64 logical channels are used or if multiple time slots are concatenated to form super channels, using one QMC channel to manage those time slots, space is freed in the dual-ported RAM. Each unused logical channel creates a 64-byte hole in the dual-ported RAM. This area is free for buffer descriptors for any SCC. QMC channels can also use this space instead of external memory for buffer descriptors, reducing the load on the external bus.

34.7 QMC Initialization

The following are the general initialization steps necessary after a reset for QMC operation:

1. Initialize QMC Global parameters, including all parameter fields, Tx and Rx time-slot assignment tables and framer tables.
2. Initialize channel-specific parameters.
3. Initialize the TxBDs and the corresponding Tx data buffers.
4. Initialize RxBDs.
5. Connect the SCC to the TDM interface through the SCC clock muxing register CMXSCR.
6. Initialize SCC registers, including enabling the SCC.
7. Program RX and TX SIRAM to point appropriate time slots to the SCC used for QMC.

8. Initialize the SI registers.
9. Enable the TDM.

34.7.1 Restarting the Transmitter

A global underrun may require the SCC transmitter to be restarted. However, for channel-specific errors, only the affected channel need be restarted. The following steps are required to restart each channel:

1. Prepare buffer descriptors.
2. Set the POL bit in the channel mode register.

A stopped, but not deactivated channel is started as described above. A deactivated channel must first have the ZISTATE and TSTATE reinitialized to their correct values, followed by setting TSATTx[V] and CHAMR[ENT]. Lastly, set CHAMR[POL] if the buffers are ready.

34.7.2 Restarting the Receiver

A global receiver overrun may require the SCC receiver to be restarted. However, for channel-specific errors, only the affected channel need be restarted. The following steps are required to restart each channel:

1. Prepare buffer descriptors.
2. Reinitialize the ZDSTATE.
3. Reinitialize the RSTATE.

34.7.3 Disabling Receiver and Transmitter

A transmit channel can be stopped from sending any more data to the line with the STOP command described in Section 34.4.1, “Transmit Commands.” The transmitter will continue to send IDLEs or FLAGs according to the channel mode register setting. To deactivate a channel, the V bit has to be cleared in the time-slot assignment table and the ENT bit has to be cleared in the channel mode register.

To stop a channel while receiving, use the STOP command as described in Section 34.4.2, “Receive Commands,” and perform a restart as described above.

Chapter 35
Universal Serial Bus Controller

The universal serial bus (USB) controller allows the MPC8555E to communicate with other devices through a USB connection. This chapter describes the MPC8555E USB controller, including its basic operation, the parameter RAM, and the registers. It also provides programming examples for initializing the host mode and function mode in the USB controller.

35.1 USB Integration in the MPC8555E

The following restrictions apply when enabling the USB controller in the MPC8555E:

- The USB controller pins are multiplexed with SCC3 pins in the parallel I/O. Refer to Chapter 45, "Parallel I/O Ports." The user programs the parallel I/O registers as if SCC3 was being used. If the USB controller is enabled, the signals are automatically routed to the USB controller instead of SCC3.
- The USB controller uses the transmit clock of SCC3 as its clock. The user must program CMXSCR[TS3CS] (refer to Section 24.4.4, "CMX SCC Clock Route Register (CMXSCR)") to the desired source for USB when the USB controller is enabled.

35.2 Overview

The USB is an industry-standard extension to the PC architecture. The USB controller on the MPC8555E supports data exchange between a wide range of simultaneously accessible peripherals. Attached peripherals share USB bandwidth through a host-scheduled, token-based protocol.

The USB physical interconnect is a tiered-star topology and the center of each star is a hub. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or a function. The USB transfers signal and power over a four-wire cable, and the signaling occurs over two wires and point-to-point segments. The USB full speed signaling bit rate is 12 Mbps. Also, a limited capability low speed signaling mode is defined at 1.5 Mbps. Refer to the *USB Specification Revision 2.0,* for further details. It can be downloaded from http://www.usb.org.

The MPC8555E USB controller consists of a transmitter module, receiver module, and two protocol state machines. The protocol state machines control the receiver and transmitter modules. One state machine implements the function state diagram and the other implements the host state diagram. The USB controller can implement a USB function endpoint, a USB host, or both for testing purposes (loopback diagnostics).

35.2.1 USB Controller Key Features

The USB function mode features are as follows:

- Four independent endpoints support control, bulk, interrupt, and isochronous data transfers
- CRC16 generation and checking
- CRC5 checking
- NRZI encoding/decoding with bit stuffing
- 12- or 1.5-Mbps data rate
- Flexible data buffers with multiple buffers per frame
- Automatic retransmission upon transmit error

The USB host controller features are as follows:

- Supports control, bulk, interrupt, and isochronous data transfers
- CRC16 generation and checking
- NRZI encoding/decoding with bit stuffing
- Supports both 12- and 1.5-Mbps data rates (automatic generation of preamble token and data rate configuration). Note that low-speed operation requires an external hub.
- Flexible data buffers with multiple buffers per frame
- Supports local loopback mode for diagnostics (12 Mbps only)

35.3 Host Controller Limitations

The following tasks are not supported by the hardware and must be implemented in software:

- CRC5 generation for tokens (because CRC5 is calculated on 11 bits, this task should not impose much software overhead)
- Retransmission after an error and error recovery
- Generation and transmission of an SOF (start of frame) token every 1 ms
- Scheduling the various transfers within and between frames

Because the MPC8555E USB host controller does not integrate the root hub, an external hub is required when more than one device is connected to the host. An external hub is also required for low-speed operation.

Also note that the host controller programming model is similar to the function endpoint programming model but does not conform to the open host controller interface (OHCI) or universal host controller interface (UHCI) standards in which software drivers are hardware-independent.

35.3.1 USB Controller Pin Functions and Clocking

The USB controller interfaces to the USB bus through a differential line driver and differential line receiver. The $\overline{\text{OE}}$ (output enable) signal enables the line driver when the USB controller transmits on the bus.

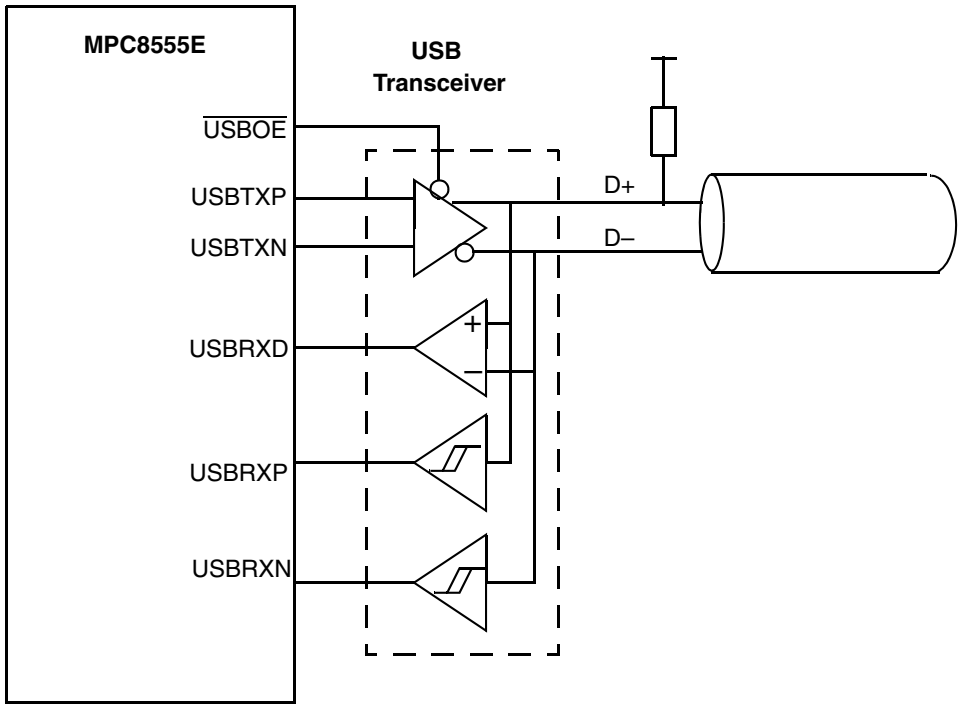


Figure 35-1. USB Interface

The reference clock for the USB controller (USBCLK) is used by the DPLL circuitry to recover the bit rate clock. The source for USBCLK is selected in CMXSCR[TS3CS] (refer to Section 24.4.4, "CMX SCC Clock Route Register (CMXSCR)"). The MPC8555E can run at different frequencies, but the USB reference clock must be four times the USB bit rate. Thus, USBCLK must be 48 MHz for a 12-Mbps full-speed transfer or 6 MHz for a 1.5-Mbps low-speed transfer.

There are six I/O pins associated with the USB port. Their functionality is described in Table 35-1. Additional control lines that might be needed by some transceivers (for example, speed select, low power control) may be supported by general purpose output lines.

Table 35-1. USB Pins Functions

Signal	I/O	Function
USBTXN, USBTXP	O	Outputs from the USB transmitter, inputs to the differential driver.
$\overline{\text{USBOE}}$	O	Output enable. Enables the transceiver to send data on the bus.

TP	TN	Result
0	0	Single Ended '0'
0	1	Logic '0'
1	0	Logic '1'
1	1	—

Table 35-1. USB Pins Functions (continued)

Signal	I/O	Function
USBRXD	I	Receive data. Input to the USB receiver from the differential line receiver.
USBRXP, USBRXN	I	Gated version of D+ and D–. Used to detect single-ended zeros and the interconnect speed.

RP	RN	Result
0	0	Single-ended ‘0’
1	0	Full speed
0	1	Low speed
1	1	—

35.4 USB Function Description

As shown in Figure 35-2, the USB function consists of transmitter and receiver sections and a control unit. The USB transmitter contains four independent FIFOs, each containing 16 bytes. There is a dedicated FIFO for each of the four supported endpoints. The USB receiver has a single 16-byte FIFO.

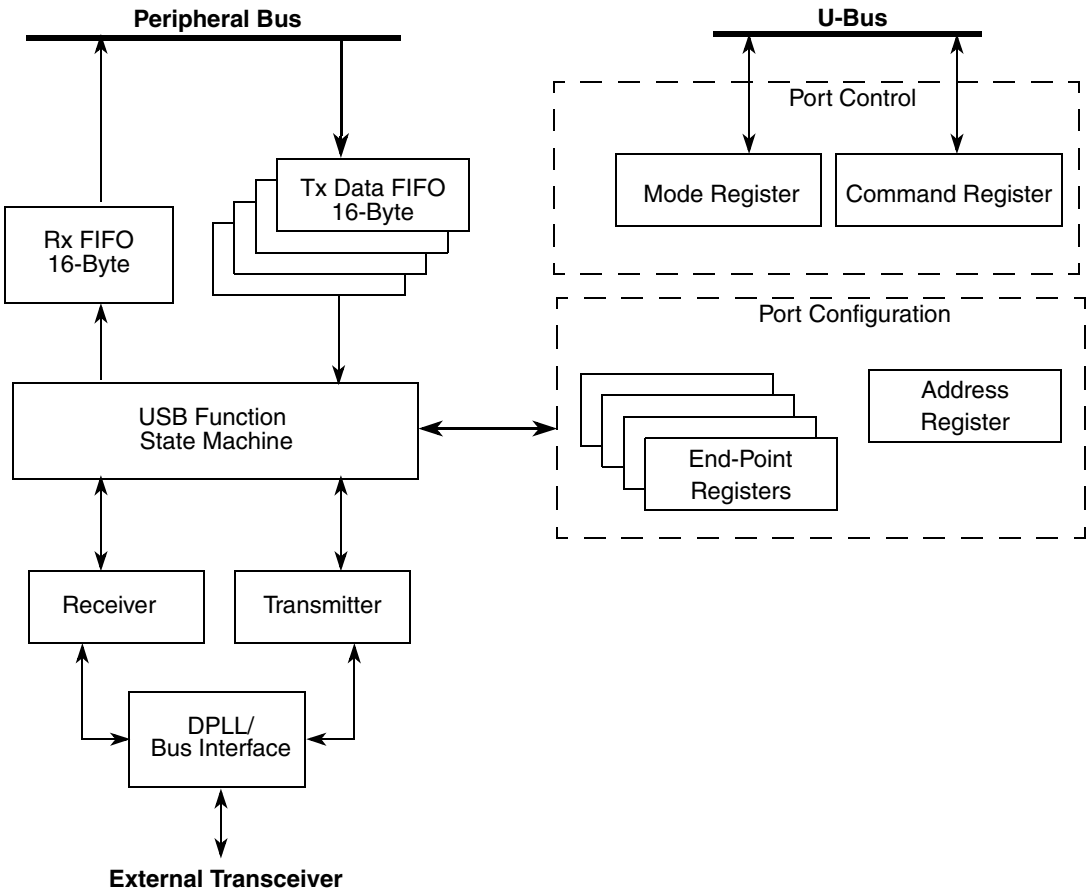


Figure 35-2. USB Function Block Diagram

35.4.1 USB Function Controller Transmit/Receive

After reset condition, the USB function is addressable at the default address (0x00). During the enumeration process the USB function is assigned by the host with a unique address. The USB slave address register (refer to Section 35.5.7.2, “USB Slave Address Register (USADR)”) should be programmed with the assigned address. The USB function controller supports four independent end-points. Each endpoint can be configured to support either control, interrupt, bulk, or isochronous transfers modes. This is done by programming the end-point registers. (Refer to Section 35.5.7.3, “USB Endpoint Registers (USEP1–USEP4).”)

NOTE

It is mandatory that endpoint 0 be configured as a control transfer type. This endpoint is used by the USB system software as a control pipe. Additional control pipes may be provided by other endpoints.

Once enabled, the USB function controller looks for valid token packets. Figure 35-3 and Table 35-2 describe the behavior of the USB controller for each token. Tokens that are not valid (that is, PID check fails or CRC check fails or packet length is not 3 bytes) are ignored by the USB function controller.

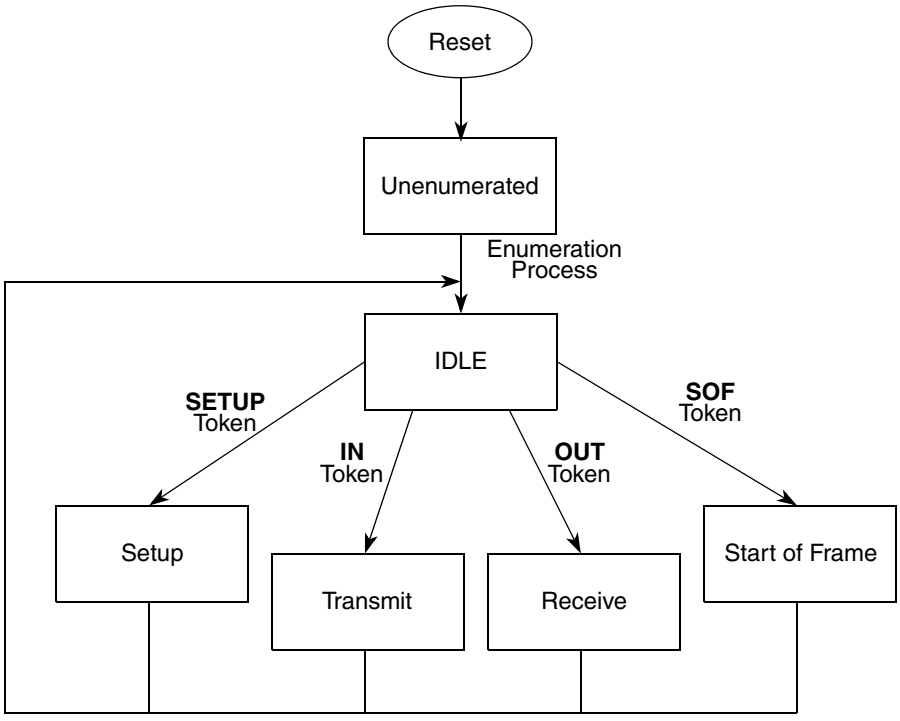


Figure 35-3. USB Controller Operating Modes

Table 35-2. USB Tokens

<table>
<tr><th>Token</th><th>Description</th></tr>
<tr><td>OUT</td><td>Reception begins when an OUT token is received. The USB controller fetches the next BD associated with the endpoint; if the BD is empty, the controller starts sending the incoming packet to the buffer. After the buffer is full, the USB controller clears RxBD[E] and generates an interrupt if RxBD[I] = 1. If the incoming packet is larger than the buffer, the USB controller fetches the next BD, and, if it is empty, sends the rest of the packet to its buffer. The entire packet, including the DATA0/DATA1 PID, are written to the receive buffers. Software must check data packet synchronization by monitoring the DATA0/DATA1 PID sequence toggle.
If the packet reception has no CRC or bit stuff errors, the USB receiver sends the handshake selected in the endpoint configuration register USEPn[RHS] (see table below) to the host. If an error occurs, no handshake packet is returned and error status bits are set in the last RxBD associated with this packet.

USB Out Token Reception
<table><tr><th>USEPn[RHS]</th><th>Data Packet Corrupted</th><th>Handshake Sent to Host</th></tr><tr><td>xx</td><td>Yes</td><td>None (data discarded)</td></tr><tr><td>00 (Normal)</td><td>No</td><td>ACK</td></tr><tr><td>01 (Ignore)</td><td>No</td><td>None</td></tr><tr><td>10 (NAK)</td><td>No</td><td>NAK</td></tr><tr><td>11 (STALL)</td><td>No</td><td>STALL</td></tr></table></td></tr>
<tr><td>IN</td><td>To guarantee a transfer, the control software must preload the endpoint FIFO with a data packet before receiving an IN token. Software should set up the endpoint TxBD table and set USCOM[STR]. The USB controller fills the transmit FIFO and waits for the IN token. Once the token is received and the FIFO has been loaded with the last data byte or with at least 4 bytes, transmission begins. The 4-byte minimum is a threshold to prevent underruns in the FIFO.
If data is not ready in the transmit FIFO or if USEPn[THS] is set to respond with NAK, a NAK handshake is returned. If USEPn[THS] was set to respond with STALL, a STALL handshake is returned. (See table below.)
When the end of the last buffer is reached (TxBD[L] is set), the CRC is appended. After the frame is sent, the USB controller waits for a handshake packet. If the host fails to acknowledge the packet, the timeout status bit TxBD[TO] is set. Software must set the proper DATA0/DATA1 PID in the transmitted packet.

USB In Token Reception
<table><tr><th>USEPn[THS]</th><th>FIFO Loaded</th><th>Handshake Sent to Host</th></tr><tr><td rowspan="2">00 (Normal)</td><td>No</td><td>NAK</td></tr><tr><td>Yes</td><td>Data packet is sent</td></tr><tr><td>01 (Ignore)</td><td>—</td><td>None</td></tr><tr><td>10 (NAK)</td><td>—</td><td>NAK</td></tr><tr><td>11 (STALL)</td><td>—</td><td>STALL</td></tr></table></td></tr>
</table>

Table 35-2. USB Tokens (continued)

Token	Description
SETUP	The format of setup transactions is similar to OUT but uses a SETUP rather than an OUT PID. A SETUP token is recognized only by a control endpoint. When a SETUP token is received, setup reception begins. The USB controller fetches the next BD associated with the endpoint; if it is empty, the controller starts transferring the incoming packet to the buffer. When the buffer is full, the USB controller clears RxBD[E] and generates an interrupt if RxBD[I] = 1. If the incoming packet is larger than the buffer, the USB controller fetches the next BD and, if it is empty, continues transferring the rest of the packet to this buffer. The entire data packet including the DATA0 PID is written to the receive buffers. If the packet was received without CRC or bit stuff errors, an ACK handshake is sent to the host. If an error occurs, no handshake packet is returned and error status bits are set in the last RxBD associated with this packet.
Start of Frame (SOF)	When an SOF packet is received, the USB controller issues a SOF maskable interrupt and the frame number entry in the parameter RAM is updated.
Preamble (PRE)	The PRE token signals the hub that a low-speed transaction is about to occur. The PRE token is read only by the hub. The USB controller ignores the PRE token function in function mode.

35.5 USB Host Description

When programmed as a host, the USB controller supports a limited host functionality. The following sections describe the available host functionality, its limitations, and the programming model.

Figure 35-4 illustrates the functionality of the USB controller in host mode. The USB controller consists of transmitter and receiver sections, host control unit, and a function control unit, which is used for testing purposes. The USB transmitter contains 4 independent FIFOs, each containing 16 bytes. Endpoint 1 is dedicated for host transactions; endpoints 2–4 are for function transactions in test mode. There is a dedicated FIFO for each of the 4 supported endpoints; endpoint 1 FIFO is for host transactions. The USB receiver has a single 16-byte FIFO.

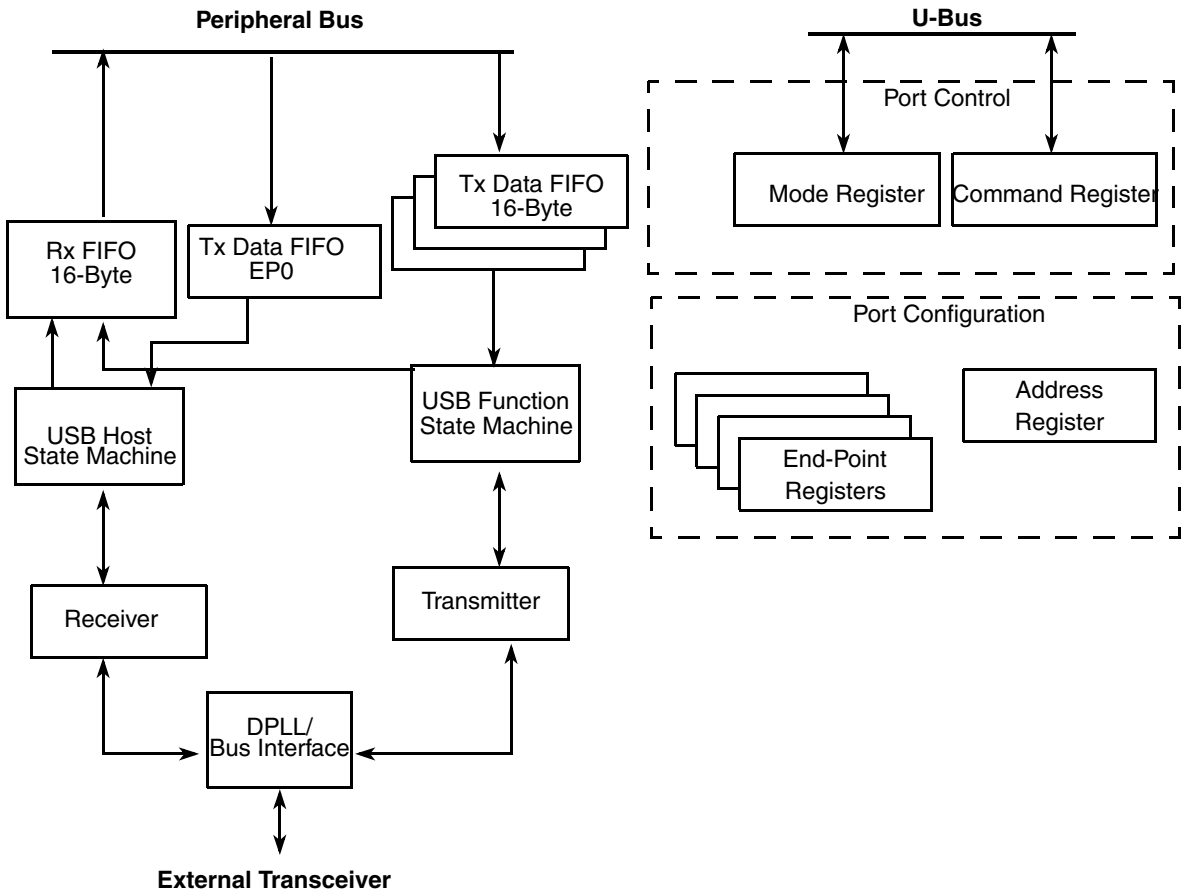


Figure 35-4. USB Controller Block Diagram

35.5.1 USB Host Controller Transmit/Receive

The USB host controller initiates all USB transactions in the system. After the reset condition, the HOST bit in USB mode register should be set (refer to Section 35.5.7.1, "USB Mode Register (USMOD),") to enable host operation. Setting USMOD[TEST] enables the loopback operation, where three of the endpoints are function endpoints. The USB controller supports four independent endpoints. Each endpoint can be configured to support either control, interrupt, bulk, or isochronous transfers modes. This is done by programming the endpoint registers. (Refer to Section 35.5.7.3, "USB Endpoint Registers (USEP1–USEP4).") Endpoint 1 must be used for host transactions (think exactly how it should be programmed and its limitations)

After reset the host should enumerate the functions in the system. The enumeration process is done by software.

Once enabled, the USB host controller waits for a packet in its FIFO. When the FIFO is filled with packet the host transaction starts. Figure 35-3 and Table 35-2 describe the behavior of the USB host controller for each token. Tokens are not checked for validity and transmitted as is. The user is responsible for token validity as well as CRC5 generation. Low speed transactions start with a preamble which is generated by the USB host controller state machine when LSP bit in token TxBD is set. The signaling on the USB lines is controlled by USMOD[LSS].

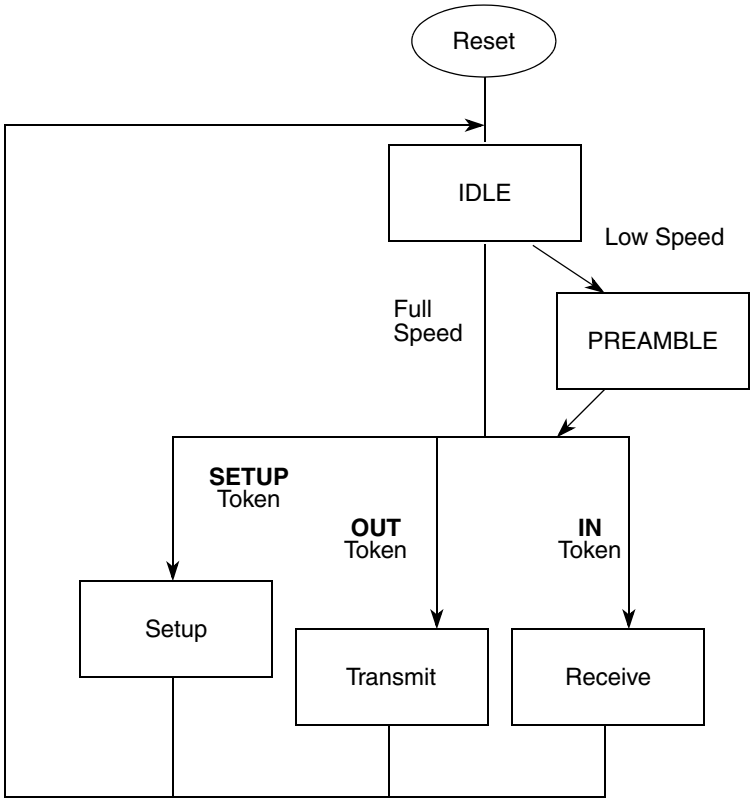


Figure 35-5. USB Controller Operating Modes

35.5.1.1 Packet-Level Interface

If USEP1[RTE] is 0, the USB host controller uses a packet-level interface to communicate with the user. Each transmit packet is prepared in a buffer and referenced by a TxBD as described in Section 35.6.3, "USB Transmit Buffer Descriptor (TxBD) for Host." Each receive packet is stored in a buffer referenced by a RxBD as described in Section 35.6.1, "USB Receive Buffer Descriptor (RxBD) for Host and Function." A SETUP or OUT transaction requires at least two TxBDs, one for the token and one or more for the data packet. An IN transaction requires one TxBD for the token and one or more RxBDs for the data packet. Tokens are not checked for validity and are transmitted as is. The user is responsible for token validity as well as CRC5 generation.

35.5.1.2 Transaction-Level Interface

If USEP1[RTE] is 1, the USB host controller uses a transaction-level interface to communicate with the user. Each transaction uses one TrDB as described in Section 35.6.4, "USB Transaction Buffer Descriptor (TrBD) for Host." The USB host controller generates the token based on the TOK field in the TrBD. For SETUP and OUT transactions, the TrBD points to a single buffer containing the data packet to be transmitted. For IN transactions, the TrBD points to a single buffer which is used for the receive data packet.

Table 35-3. USB Tokens

<table>
<tr><th>Token</th><th colspan="2">Description</th></tr>
<tr><td>OUT</td><td>Packet-Level Interface

Transmission begins when the USB host controller fetches a TxBD containing an OUT token and a data TxBD and loads them to the host FIFO. The token and data are transmitted and a handshake is expected.
If a handshake is not received within the expected time interval, the USB controller clears TxBD[R] of data BD, sets the TxBD[TO] indication and generates a TXE1 interrupt.
When STALL or NAK is received within the expected time interval, the USB controller clears TxBD[R] of data BD, sets the TxBD[STALL] or TXBD[NAK] indication and generates a TXE1 interrupt. When ACK is received within the expected time interval, the USB controller clears TxBD[R] of data BD, and generates an interrupt if TxBD[I] = 1. No indication is set.
The token TxBD[R] is cleared right after the OUT token transmission.</td><td>Transaction-Level Interface

Transmission begins when the USB host controller fetches a TrBD with the TOK field indicating an OUT transaction. The token is generated and then the data packet from the buffer is transmitted and a handshake is expected.
If a handshake is not received within the expected time interval, the USB controller clears TrBD[R], sets the TrBD[TO] indication and generates a TXE1 interrupt.
When STALL or NAK is received within the expected time interval, the USB controller clears TrBD[R], sets the TrBD[STALL] or TrBD[NAK] indication and generates a TXE1 interrupt. When ACK is received within the expected time interval, the USB controller clears TrBD[R], and generates a TXB interrupt if TrBD[I] = 1.No indication is set.</td></tr>
<tr><td></td><td colspan="2">USB Out Transaction
<table>
<tr><th>Token</th><th>Data</th><th>Handshake Received by Host</th><th>Indication on TxBD/TrBD</th></tr>
<tr><td rowspan="4">OUT</td><td rowspan="4">Sent by host</td><td>None</td><td>TO</td></tr>
<tr><td>ACK</td><td>None</td></tr>
<tr><td>NAK</td><td>NAK</td></tr>
<tr><td>STALL</td><td>STALL</td></tr>
</table></td></tr>
</table>

Table 35-3. USB Tokens (continued)

<table>
<tr><th>Token</th><th colspan="2">Description</th></tr>
<tr><td>IN</td><td>Packet-Level Interface

Transmission begins when the USB host controller fetches a TxBD containing an IN token and loads the token to FIFO. After the IN token is transmitted the USB host controller waits for reception of data within expected time interval. On reception of a valid DATA PID an RxBD is fetched. The received data and DATA PID are stored in receive FIFO. If RxBD[E] is set PID and data will be moved to the buffer. While receiving the data the USB host controller calculates CRC16, performs bit un-stuffing. On end of reception calculated CRC is compared to received and octet alignment is checked, RxBD[E] is cleared, RxBD[PID] is set according to received DATA PID and error indications are set if required:RxBD[CR] for failed CRC check, RxBD[NO] for non-octet sized data and RxBD[AB] if bit stuffing error occurred.
If no valid DATA PID or no data at all received during the expected time interval a TO indication in the token TxBD is set.</td><td>Transaction-Level Interface

Transmission begins when the USB host controller fetches a TrBD with the TOK field indicating an IN transaction. After the IN token is generated and transmitted, the USB host controller waits for reception of data within the expected time interval. The received data packet is stored in buffer reference by the TrBD. While receiving the data the USB host controller calculates CRC16 and performs bit un-stuffing. At end of the packet, the calculated CRC is compared to the received value and octet alignment is checked, TrBD[R] is cleared, TrBD[PID] is set according to the received DATA PID and error indications are set if required: TrBD[CR] for failed CRC check, TrBD[NO] for non-octet sized data and TrBD[AB] if bit stuffing error occurred. If any of the above errors are reported, TrBD[RXER] is also set, and a TXE1 interrupt is generated.
If no valid DATA PID or no data at all received during the expected time interval, a TrBD[TO] is set and a TXE1 interrupt is generated.
If no errors occurred and TrBD[I] is set, a TXB interrupt is generated to indicate successful completion of the transaction.</td></tr>
<tr><td></td><td colspan="2">USB In Transaction
<table>
<tr><th>Token</th><th>Data Transmitted by Function</th><th>Handshake Generated by Host</th><th>Indication on BD</th></tr>
<tr><td rowspan="3">IN</td><td>Received correctly</td><td>ACK</td><td>RxBD[E]/TrBD[R] is cleared</td></tr>
<tr><td>Received corrupted</td><td>None</td><td>RxBD[CR]/TrBD[CR} or RxBD[AB]/TrBD[AB] or RxBD[NO]/TrBD[NO]</td></tr>
<tr><td>None</td><td>None</td><td>TxBD[TO]/TrBD[TO]</td></tr>
</table></td></tr>
<tr><td>SETUP</td><td colspan="2">The format of setup transactions is similar to OUT but uses a SETUP rather than an OUT PID. A SETUP token is recognized only by a control endpoint. When a SETUP token is received, setup reception begins. The USB controller fetches the next BD associated with the endpoint; if it is empty, the controller starts transferring the incoming packet to the buffer. When the buffer is full, the USB controller clears RxBD[E] and generates an interrupt if RxBD[I] = 1. If the incoming packet is larger than the buffer, the USB controller fetches the next BD and, if it is empty, continues transferring the rest of the packet to this buffer. The entire data packet including the DATA0 PID is written to the receive buffers. If the packet was received without CRC or bit stuff errors, an ACK handshake is sent to the host. If an error occurs, no handshake packet is returned and error status bits are set in the last RxBD associated with this packet.</td></tr>
<tr><td>Start of Frame (SOF)</td><td colspan="2">When an SOF packet is received, the USB controller issues a SOF maskable interrupt and the frame number entry in the parameter RAM is updated.</td></tr>
<tr><td>Preamble (PRE)</td><td colspan="2">The PRE token signals the hub that a low-speed transaction is about to occur. The PRE token is read only by the hub. The USB controller ignores the PRE token function in function mode.</td></tr>
</table>

35.5.2 SOF Transmission for USB Host Controller

The following section describes the mechanism used by the USB host controller to support the automatic transmission of SOF tokens. This mechanism is enabled by setting USMOD[SFTE].

SOF packets should be transmitted every 1 ms. Since the time interval between two SOF packets must be more precise than could be accomplished by software, a hardware timer is used to assert an interrupt to the CP. When the interrupt is serviced by the CP, it prepares a SOF token and loads it to the host endpoint. Once the SOF token is loaded to the FIFO, it is transmitted like any other packet.

Before each SOF transmission, the software should prepare a value for the frame number and CRC5 to be transmitted in SOF token and place it in the parameter RAM (for further details refer to Section 35.5.5, "Frame Number (FRAME_N)." One possible implementation would be to use the SOF interrupt (see Section 35.5.7.5, "USB Event Register (USBER)") to prepare the frame number for the next SOF packet. The SFT interrupt should not be used for this purpose since it is generated before the SOF packet is actually transmitted.

The application software should also guarantee that the USB host has completed all pending transactions prior to the 1 ms tick, so that the transmit FIFO is empty at this point. The current value of the SOF timer may be read at any time to synchronize the software with the USB frames. See Section 35.5.7.8, "USB Start of Frame Timer (USSFT)," for more information.

35.5.3 USB Function and Host Parameter RAM Memory Map

The USB controller parameter RAM area, shown in Table 35-4, begins at the USB base address, 0x8B00 (offset from RAM_Base). Note that the user must initialize certain parameter RAM values before the USB controller is enabled.

Table 35-4. USB Parameter RAM Memory Map

Address	Name[1]	Width	Description
USB Base + 00	**EP0PTR**	Half word	Endpoint pointer registers 0–3. The endpoint parameter block pointers are index pointers to each endpoint's parameter block. Parameter blocks can be allocated to any address divisible by 32 in the dual port RAM. See Figure 35-6. The map of the endpoint parameter block is shown in Table 35-5. **Note**: When USB host mode is set **EP0PTR** must be used for the host endpoint.
USB Base + 02	**EP1PTR**	Half word	
USB Base + 04	**EP2PTR**	Half word	
USB Base + 06	**EP4PTR**	Half word	
USB Base + 08	RSTATE	Word	Receive internal state. Reserved for CP use only. Should be cleared before enabling the USB controller.
USB Base + 0C	RPTR	Word	Receive internal data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
USB Base + 10	**FRAME_N**	Half word	Frame number. See Figure 35-7. **Note**: The definition of this parameter is different for host mode and function mode.
USB Base + 12	RBCNT	Half word	Receive internal byte count. A down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.
USB Base + 14	RTEMP	Word	Receive temp. Reserved for CP use only.

Table 35-4. USB Parameter RAM Memory Map (continued)

Address	Name[1]	Width	Description
USB Base + 18	RXUSB_ Data	Word	Rx data temp
USB Base + 1C	RXUPTR	Half word	Rx microcode return address temp

1 The items in **boldface** should be initialized by the user before the USB controller is enabled; other values are initialized by the CP.

Once initialized, the parameter RAM values do not normally need to be accessed by user software. They should only be modified when no USB activity is in progress.

35.5.4 Endpoint Parameters Block Pointer (EP*n*PTR)

The endpoint parameter block pointers (EP*n*PTR) are DPRAM in indices to an endpoint's parameter block. The parameter block can be allocated to any address that is divisible by 32. The format of the endpoint pointer registers (EP*n*PTR) is shown in Figure 35-6.

	0 – 10	11 – 15
Field	Endpoint Index Pointer	—
R/W	R/W	
Reset	0000_0000_0000_0000	
Offset	USB base + 0x00 (EP0PTR), 0x02 (EP1PTR), 0x04 (EP2PTR), 0x06 (EP3PTR)	

Figure 35-6. Endpoint Pointer Registers (EP*n*PTR)

The map of the endpoint parameter block is shown in Table 35-5.

Table 35-5. Endpoint Parameter Block

Offset[1]	Name[2]	Width	Description
0x00	**RBASE**	16 bits	RxBD/TxBD base addresses. Define the starting location in dual-port RAM for the USB controller's TxBDs and RxBDs. This provides flexibility in how BDs are partitioned. Setting W in the last BD in each list determines how many BDs to allocate for the controller's send and receive sides. These entries must be initialized before the controller is enabled. Overlapping USB BD tables with another serial controller's BDs causes erratic operation. RBASE and TBASE values should be divisible by 8.
0x02	**TBASE**	16 bits	
0x04	**RFCR**	8 bits	Rx/Tx function code. Controls the value to appear on AT[1:3] when the associated SDMA channel accesses memory and the byte-ordering convention.
0x05	**TFCR**	8 bits	

Table 35-5. Endpoint Parameter Block (continued)

Offset[1]	Name[2]	Width	Description
0x06	**MRBLR**	16 bits	Maximum receive buffer length. Defines the maximum number of bytes the MPC8555E writes to the USB receive buffer before moving to the next buffer. MRBLR must be divisible by 4. The MPC8555E can write fewer data bytes to the buffer than the MRBLR value if a condition such as an error or end-of-packet occurs, but it never exceeds MRBLR. Therefore, user-supplied buffers should never be smaller than MRBLR. MRBLR is not designed to be changed dynamically for the currently active RxBD during USB operation; however, MRBLR can be modified safely for the next and subsequent RxBDs using a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). Transmit buffers for the USB controller are not affected by the MRBLR value. Transmit buffer lengths can vary individually, as needed. The number of bytes to be sent is chosen by programming TxBD[Data Length].
0x08	**RBPTR**	16 bits	RxBD pointer. Points to the next BD the receiver will transfer data to when it is in an idle state or to the current BD while processing a frame. Software should initialize RBPTR after reset. When the end of the BD table is reached, the CP initializes this pointer to the value programmed in RBASE. Although the user does not need to write RBPTR in most applications (except initialization), it can be changed when the receiver is disabled or when no receive buffer is being used.
0X0A	**TBPTR**	16 bits	TxBD pointer. Points to the next BD that the transmitter will transfer data from when it is in an idle state or to the current BD during frame transmission. TBPTR should be initialized by the software after reset. When the end of BD table is reached, the CP initializes this pointer to the value programmed in the TBASEn entry. Although the user never needs to write TBPTR, in most applications (except initialization), it can be changed when the transmitter is disabled or when no transmit buffer is being used.
0X0C	TSTATE[3]	32 bits	Transmit internal state. Reserved for CP use only. Should be cleared before enabling the USB controller.
0x10	TPTR[3]	32 bits	Transmit internal data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x14	TCRC[3]	16 bits	Transmit temp CRC. Reserved for CP use only.
0x16	TBCNT[3]	16 bits	Transmit internal byte count. A down-count value that is initialized with the TxBD data length and decremented with every byte read by the SDMA channels.
0x18	TTEMP	32 bits	Tx temp
0x1C	TXUSBU_PTR	16 bits	Tx microcode return address temp
0x1E	**HIMMR**	16 bits	When using the transaction-based interface in host mode, this field must be programmed to {CCSRBAR[0:11], 0x9}. Otherwise, this field is unused.

1 Offset from endpoint parameter block base.

2 Note that the items in **boldface** should be initialized by the user.

3 These parameters need not be accessed in normal operation but may be helpful for debugging.

35.5.5 Frame Number (FRAME_N)

This entry is used for frame number updates both in function mode and in host mode. In function mode it is updated by the USB controller; in host mode it is updated by the application software.

This entry is updated by the USB controller in function mode whenever a SOF (start of frame) token is received. The entry contains 11 bits that represent the frame number. An SOF interrupt is issued upon an update of this entry.

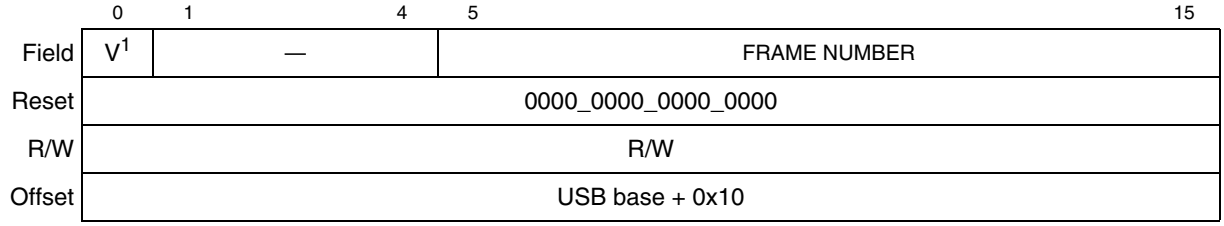


Figure 35-7. Frame Number (FRAME_N) in Function Mode—Updated by USB Controller

[1] This bit is set if the SOF token was received error free.

Table 35-6 describes FRAME_N fields.

Table 35-6. FRAME_N Field Descriptions

Bits	Name	Description
0	V	The valid bit is set if the SOF token is received without error.
1–4	—	Reserved, should be cleared.
5–15	FRAME NUMBER	The frame number is loaded with the value received in the SOF packet. Be sure the frame number is cleared before beginning USB operation.

This entry is updated by the application software whenever a SOF (start of frame) token should be received. The software should prepare the frame number and the CRC and place it in FRAME_N field.

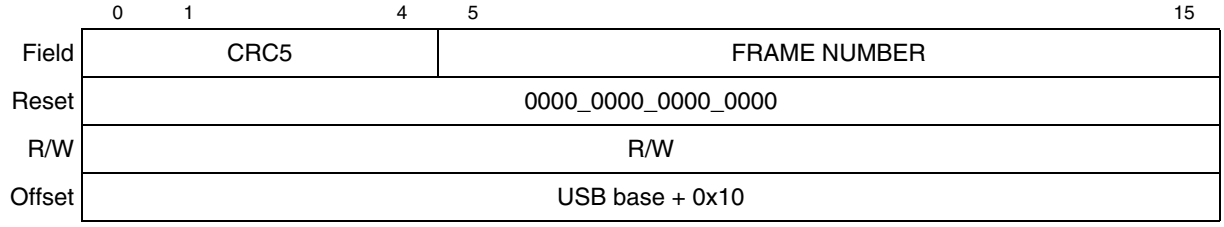


Figure 35-8. Frame Number (FRAME_N) in Function Mode—Updated by Application Software

Table 35-6 describes FRAME_N fields.

Table 35-7. FRAME_N Field Descriptions

Bits	Name	Description
0–4	CRC5	CRC5 calculated on frame number
5–11	FRAME NUMBER	The frame number is inserted by the application software.

NOTE

The FRAME NUMBER field is also updated by the USB controller when the USB controller is configured as the host, thus indicating that SOF was transmitted. Therefore, the FRAME NUMBER field should always be regenerated and rewritten to the entry before SOF is issued.

35.5.6 USB Function Code Registers (RFCR and TFCR)

RFCR and TFCR control the value that the user would like to appear on the Address Type pins (AT1–AT3) when the associated SDMA channel accesses memory.

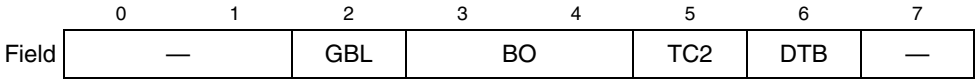


Figure 35-9. USB Function Code Registers (RFCR and TFCR)

Table 35-8 describes RFCR and TFCR fields.

Table 35-8. RFCR and TFCR Fields

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering. This bit field should be set by the user to select the required byte ordering for the data buffer. If this bit field is modified on-the-fly, it will take effect at the beginning of the next frame. 00 DEC (and Intel) convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The transmission order of bytes within a buffer word is reversed as compared to the big-endian mode. This mode is supported only for 32-bit port size memory. 01 PowerPC little-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the least significant byte of the buffer double-word contains data to be transmitted earlier than the most significant byte of the same buffer double word. 1X Big-endian byte ordering—normal operation. As data is transmitted onto the serial line from the data buffer, the most significant byte of the buffer word contains data to be transmitted earlier than the least significant byte of the same buffer word.
5	TC2	Transfer code. Contains the transfer code value of TC[2] used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access
6	DTB	Data bus indicator 0 Use system bus for SDMA operation 1 Use local bus for SDMA operation
7	—	Reserved, should be cleared.

35.5.7 USB Function Programming Model

The following sections describe USB controller registers.

35.5.7.1 USB Mode Register (USMOD)

USMOD controls the USB controller operation mode.

	0	1	2	3	4	5	6	7
Field	LSS	RESUME	—		SFTE	TEST	HOST	EN
Reset	0000_0000							
R/W	R/W							
Offset	0x9_1B60							

Figure 35-10. USB Mode Register (USMOD)

Table 35-9 describes USMOD fields.

Table 35-9. USMOD Fields

Bits	Name	Description
0	LSS	Low-speed signaling. Selects the signaling speed. The actual bit rate depends on the USB clock source. 0 Full-speed (12-Mbps) signaling. Normal operation. 1 Low-speed (1.5-Mbps) signaling. For a point-to-point connection with a low-speed device or for local loopback testing.
1	RESUME	Generate resume condition. When set, this bit generates a resume condition on the USB. This bit should be used if the function wants to exit the suspend state.
2–3	—	Reserved, should be cleared.
4	SFTE	Start-of-frame timer enable. Setting this bit enables the Start-of-Frame timer and automatic SOF transmission. See Section 35.5.7.8, “USB Start of Frame Timer (USSFT),” and Section 35.5.2, “SOF Transmission for USB Host Controller,” for more information. 0 SOF timer is disabled 1 SOF timer is enabled **Note:** When SFTE is 1, the PC21 pin cannot be used as CP_INT since the CP interrupt is used internally for generating the SOF packet.
5	TEST	USB controller test(loopback) mode 0 Test mode is disabled 1 Test mode is enabled **Note:** This bit may be set only when HOST is set (USB host mode)
6	HOST	USB host mode 0 USB host disabled 1 USB host is enabled
7	EN	Enable USB. When the EN bit is cleared, the USB is in a reset state 0 USB is disabled 1 USB is enabled **Note:** Setting this bit automatically disables SCC3. **Note:** Other bits of the USMOD should not be modified by the user while EN is set.

35.5.7.2 USB Slave Address Register (USADR)

The USB address register is an 8-bit, memory-mapped register. It holds the address for this USB port when operating as function.

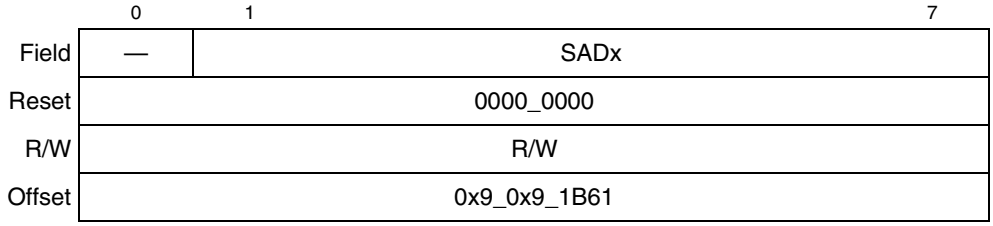


Figure 35-11. USB Slave Address Register (USADR)

Table 35-10 describes USADR fields.

Table 35-10. USADR Fields

Bits	Name	Description
0	—	Reserved, should be cleared.
1–7	SADx	Slave address 0–6. Holds the slave address for the USB port, when configured as function.

35.5.7.3 USB Endpoint Registers (USEP1–USEP4)

There are four memory-mapped endpoint configuration registers.

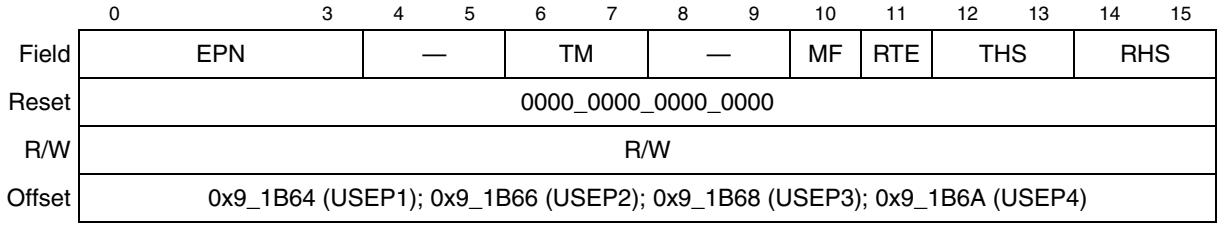


Figure 35-12. USB Endpoint Registers (USEP1–USEP4)

Table 35-11 describes the fields of USEP1–USEP4. The setting for USB host controller should be set only in USEP1, when USMOD[HOST] is set.

Table 35-11. USEP*n* Field Descriptions

Bits	Name	USB Function Mode	USB Host Mode
0–3	EPN	Endpoint number. For USB **function** controller defines the supported endpoint number.	For USB **host** controller, should be cleared.
4–5	—	Reserved, should be cleared.	Reserved, should be cleared.
6–7	TM	Transfer mode for USB **function** controller 00 Control 01 Interrupt 10 Bulk 11 Isochronous	Transfer mode for USB **host** controller 00 Control /interrupt/bulk 11 Isochronous
8–9	—	Reserved, should be cleared.	Reserved, should be cleared

Table 35-11. USEP*n* Field Descriptions (continued)

Bits	Name	USB Function Mode	USB Host Mode
10	MF	Enable multi-frame. For USB **function** controller allows loading of the next transmit packet into the FIFO before transmission completion of the previous packet. 0 Transmit FIFO may hold only one packet 1 Transmit FIFO may hold more than one packet **Note:** For USB function configuration: Should be cleared unless the endpoint is configured for ISO transfer mode.	Enable multi-frame for USB **host** controller. Should be always set.
11	RTE	Retransmit enable for USB **function** controller 0 No retransmission 1 Automatic frame retransmission is enabled. The frame will be retransmitted if transmit error occurred (time-out). **Note:** May be set only if the transmit packet is contained in a single buffer. If it is not, retransmission should be handled by software intervention. **Note:** Should be set to zero for endpoint which is configured for ISO transfer mode	For USB **host** controller, should be cleared.
12–13	THS	Transmit handshake for USB **function** controller 00 Normal handshake 01 Ignore IN token 10 Force NACK handshake. Not allowed for control endpoint. 11 Force STALL handshake. Not allowed for control endpoint.	Transmit handshake for USB **host** controller 00 Normal handshake
14–15	RHS	Receive handshake for USB **function** controller 00 Normal handshake 01 Ignore OUT token 10 Force NACK handshake. Not allowed for control endpoint. 11 Force STALL handshake. Not allowed for control endpoint.	Receive handshake for USB **host** controller 00 Normal handshake

35.5.7.4 USB Command Register (USCOM)

USCOM is used to start USB transmit operation.

	0	1	2	3	4	5	6	7
Field	STR	FLUSH	ISFT	DSFT	—		EP	
Reset	0000_0000							
R/W	R/W							
Offset	0x9_1B62							

Figure 35-13. USB Command Register (USCOM)

Table 35-12 describes USCOM fields.

Table 35-12. USCOM Fields

Bits	Name	Description
0	STR	Start FIFO fill. Setting the STR bit to one causes the USB controller to start the filling the corresponding endpoint transmit FIFO with data. Transmission will begin once the IN token for this end-point is received. The STR bit is read always as a zero.
1	FLUSH	Flush FIFO. Setting the FLUSH bit to one causes the USB controller to flush the corresponding endpoint transmit FIFO. Before flushing the FIFO, the user should issue the Stop_Tx command. After flushing the FIFO the user should issue the Restart_Tx command. (Refer to Section 35.7, “USB CP Commands.”) FLUSH is always read as a zero.
2	ISFT	Increment start-of-frame time. Setting the ISFT bit increments the start-of-frame time by one. This bit could be used to synchronize the USB frames to an external timing source.
3	DSFT	Decrement start-of-frame time. Setting the DSFT bit decrements the start-of-frame time by one. This bit could be used to synchronize the USB frames to an external timing source.
4–5	—	Reserved, should be cleared.
6–7	EP	Endpoint. Selects one of the four supported endpoints.

35.5.7.5 USB Event Register (USBER)

The USBER reports events recognized by the USB channel and generates interrupts. Upon recognition of an event, the USB sets its corresponding bit in the USBER. Interrupts generated by this register may be masked in the USB mask register.

The USBER may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit’s value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

	0–4	5	6	7	8	9	10	11	12	13	14	15
Field	—	SFT	RESET	IDLE	TXE4	TXE3	TXE2	TXE1	SOF	BSY	TXB	RXB
Reset	0000_0000_0000_0000											
R/W	R/W											
Offset	0x9_1B70											

Figure 35-14. USB Event Register (USBER)

Table 35-13 describes USBER fields.

Table 35-13. USBER Fields

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	SFT	The start-of-frame timer (USSFT[SFT]) wrapped from 11,999 to 0.
6	RESET	Reset condition detected. USB reset condition was detected asserted.
7	IDLE	IDLE status changed. A change in the status of the serial line was detected. The real time suspend status is reflected in the USB status register.

Table 35-13. USBER Fields (continued)

Bits	Name	Description
8–11	TXEx	Tx error. An error occurred during transmission for Endpoint x (packet not acknowledged or underrun).
12	SOF	Start-of-frame. A start of frame packet was received. The packet is stored in the FRAME_N parameter ram entry.
13	BSY	Busy condition. Received data has been discarded due to a lack of buffers. This bit is set after the first character is received for which there is no receive buffer available.
14	TXB	Tx buffer. A buffer has been transmitted. This bit is set once the transmit data of the last character in the buffer was written to the transmit FIFO (if L = 0 (last bit)) or after the last character was transmitted on the line (if L = 1).
15	RXB	Rx buffer. A buffer has been received. This bit is set after the last character has been written to the receive buffer and the RxBD is closed.

35.5.7.6 USB Mask Register (USBMR)

The USBMR is a 16-bit read/write register (0x9_1B74) that has the same bit formats as the USB event register. If a bit in the USBMR is one, the corresponding interrupt in the USBER is enabled. If the bit is zero, the corresponding interrupt in the USBER will be masked. This register is cleared at reset.

35.5.7.7 USB Status Register (USBS)

The USB status register, described in Figure 35-15 and Table 35-14, is a read-only register that allows the user to monitor real-time status condition on the USB lines.

	0 — 6	7
Field	—	IDLE
Reset	0000_0000	
R/W	R	
Offset	0x9_1B77	

Figure 35-15. USB Status Register (USBS)

Table 35-14 describes USBS fields.

Table 35-14. USBS Fields

Bits	Name	Description
0–6	—	Reserved
7	IDLE	Idle status. IDLE is set when an idle condition is detected on the USB lines, it is cleared when the bus is not idle.

35.5.7.8 USB Start of Frame Timer (USSFT)

When enabled by USMOD[SFTE], the USSFT contains the current time within the frame with a resolution of one bit time. When the value of USSFT wraps from 11,999 to 0, a CP interrupt is asserted to trigger the transmission of a SOF packet, and USBER[SFT] is set.

The USSFT may be read at any time.

	0	1	2 ... 15
Field	—		SFT
Reset	0010_1110_1101_1000		
R/W	R		
Addr	0x9_1B78		

Figure 35-16. USB Start of Frame Timer (USSFT)

Table 35-15 describes USSFT fields.

Table 35-15. USSFT Fields

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–15	SFT	Start-of-frame time. This field contains the number of bit times since the last SOF trigger. Note that the actual SOF transmission occurs slightly later.

35.6 USB Buffer Descriptor Ring

The data associated with the USB channel is stored in buffers that are referenced by BDs organized in BD rings located in the dual-port RAM (refer to Figure 35-17). These rings have the same basic configuration as those used by the SCCs and SMCs.

There are four separate transmit BD rings and four separate receive BD rings, one for each endpoint. The BD ring allows the user to define buffers for transmission and buffers for reception. Each BD ring forms a circular queue. The CP confirms reception and transmission or indicates error conditions using the BDs to inform the processor that the buffers have been serviced.

The buffers may reside in either external or internal memory.

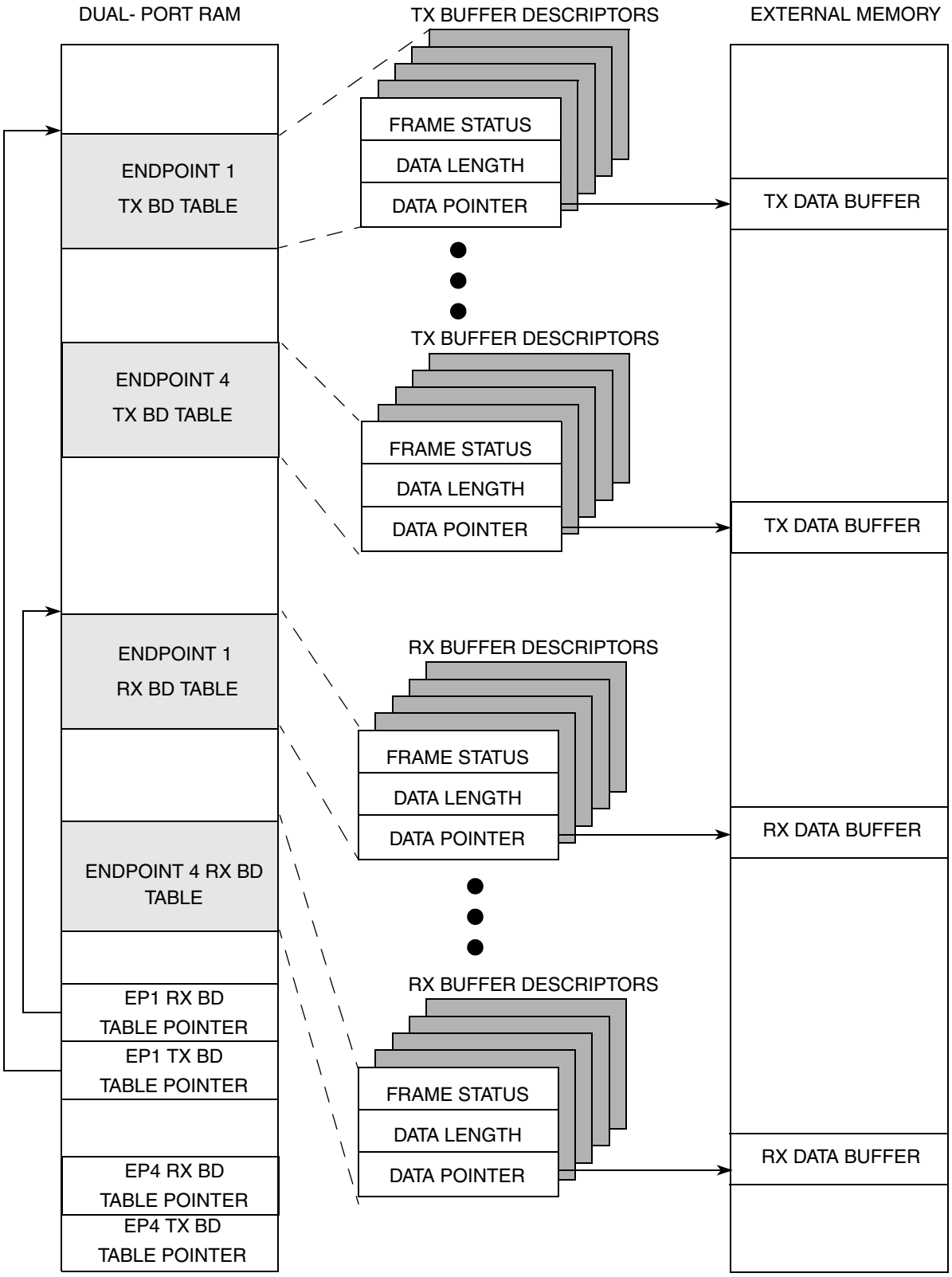


Figure 35-17. USB Memory Structure

35.6.1 USB Receive Buffer Descriptor (RxBD) for Host and Function

The CP reports information about each buffer of received data using RxBDs. The CP closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer when the current buffer is full. Additionally, it closes the buffer on the following conditions:

- End of packet detected
- Overrun error occurred
- Bit stuff violation detected

As shown in Figure 35-18, the first word of the RxBD contains status and control bits. These bits are prepared by the user before reception and are set by the CP after the buffer has been closed. The second word contains the data length—in bytes—that was received. The third and fourth words contain a pointer that always points to the beginning of the received data buffer.

The RxBD is identical for both the host mode (when using the packet-level interface) and the function mode. There are no RxBDs in host mode when using the transaction-level interface.

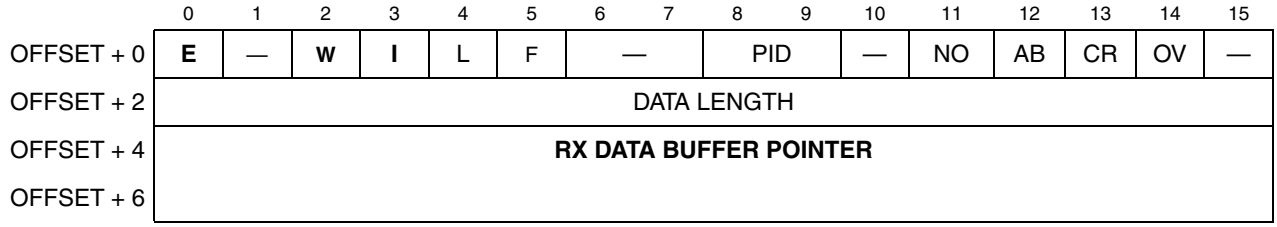


Figure 35-18. USB Receive Buffer Descriptor (RxBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be written by the CPU core before enabling the USB

Table 35-16 describes USB receive buffer descriptor fields.

Table 35-16. USB RxBD Fields

Offset	Bits	Name	Description
0x00	0	**E**	Empty 0 The data buffer associated with this RxBD has been filled with received data, or data reception has been aborted due to an error condition. The CPU core is free to examine or write to any fields of this RxBD. The CP will not use this BD again while the E-bit remains zero. 1 The data buffer associated with this BD is empty, or reception is currently in progress. This RxBD and its associated receive buffer are owned by the CP. Once the E-bit is set, the CPU core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table (the BD pointed to by RBASE). The number of RxBDs in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

Table 35-16. USB RxBD Fields (continued)

Offset	Bits	Name	Description
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been filled. 1 The RXB bit in the USB event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU core to process the buffer. The RXB bit can cause an interrupt if it is enabled.
	4	L	Last. This bit is set by the USB controller when the buffer is closed due to detection of end-of-packet condition on the bus, or as a result of error. Written by the USB controller after the received data has been placed into the associated data buffer. 0 Buffer does not contain the last byte of the message. 1 Buffer contains the last byte of the message.
	5	F	First. This bit is set by the USB controller when the buffer contains the first byte of a packet. Written by the USB controller after the received data has been placed into the associated data buffer. 0 Buffer does not contain the first byte of the message. 1 Buffer contains the first byte of the message.
	6–7	—	Reserved, should be cleared.
	8–9	PID	Packet ID. This bit field is set by the USB controller to indicate the type of the packet. This bit is valid only if the USB RXBD[F] is set. Written by the USB controller after the received data has been placed into the associated data buffer. 00 Buffer contains DATA0 packet 01 Buffer contains DATA1 packet 10 Buffer contains SETUP packet. This option can never be set on host RxBD.
	10	—	Reserved, should be cleared.
	11	NO	Rx non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data has been placed into the associated data buffer.
	12	AB	Frame aborted. Bit stuff error occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	13	CR	CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer. Written by the USB controller after the received data has been placed into the associated data buffer.
	14	OV	Overrun. A receiver overrun occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	15	—	Reserved, should be cleared.
0x02	0–15	Data length	Data length is the number of octets that the CP has written into this BD's data buffer. It is written once by the CP as the BD is closed. **Note:** The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.
0x04	0–31	**Rx data buffer pointer**	The receive buffer pointer, which always points to the first location of the associated data buffer, must be divisible by 4. The buffer may reside in either internal or external memory

Data length represents the number of octets that the CP has written into this BD's buffer. It is written once by the CP as the BD is closed.

The receive buffer pointer always points to the first location of the associated buffer. The pointer must be divisible by four. The buffer may reside in either internal or external memory.

35.6.2 USB Transmit Buffer Descriptor (TxBD) for Function

Data that the USB function wishes to transmit to the host is arranged in buffers referenced by the TxBD ring. The first word of the TxBD contains the status and control bits.

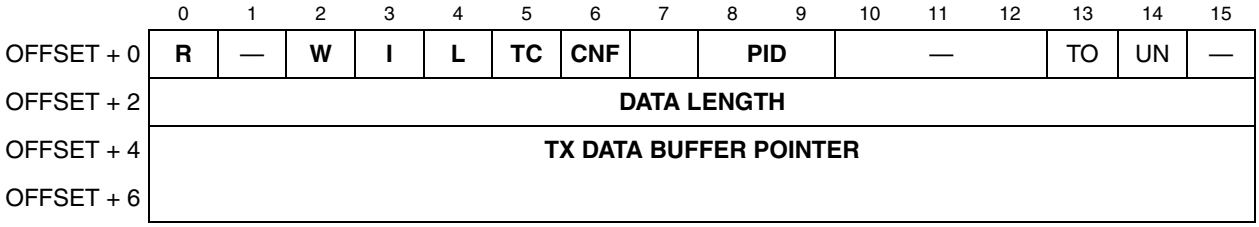


Figure 35-19. USB Transmit Buffer Descriptor (TxBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 35-17 describes USB TxBD fields.

Table 35-17. USB Function TxBD Fields

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASEx). The number of TxBDs in this table is programmable, and is determined only by the TxBD[W] and the overall space constraints of the dual-port RAM.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 The TXB or TXE bit in the event register is set when this buffer is serviced. TXB and TXE can cause interrupts if they are enabled.
	4	**L**	Last 0 Buffer does not contain the last byte of the message 1 Buffer contains the last byte of the message
	5	TC	Transmit CRC. Valid only when the L bit is set; otherwise it is ignored. Prepare TC before sending data. 0 Transmit end-of-packet after the last data byte. This setting can be used for testing purposes to send a bad CRC after the data. 1 Transmit the CRC sequence after the last data byte.

Table 35-17. USB Function TxBD Fields (continued)

Offset	Bits	Name	Description
	6	CNF	Transmit confirmation. Valid only when the L bit is set; otherwise it is ignored. Applies to multi-frame enabled endpoints (USEP*n*[MF] = 1); refer to Section 35.5.7.3, "USB Endpoint Registers (USEP1–USEP4)." 0 Continue to load the transmit FIFO with the next packet. Several packets may be loaded to the FIFO. 1 Last packet that is loaded to FIFO. No more packets will be loaded to fifo after a packet marked CNF, till it transmitted.
	7		Reserved, should be cleared
	8–9	**PID**	Packet ID. This bit field is valid for the first BD of a packet; otherwise it is ignored. 0X Do not append PID to the data. 10 Transmit DATA0 PID before sending the data. 11 Transmit DATA1 PID before sending the data.
	10–12	—	Reserved, should be cleared.
	13	TO	Time out. Indicates that the host failed to acknowledge the packet.
	14	UN	Underrun. Indicates that the USB encountered a transmitter underrun condition while sending the buffer.
	15	—	Reserved, should be cleared.
0x02	0–15	**Data length**	The data length is the number of octets that the CP should transmit from this BD's data buffer. It is never modified by the CP. This value should normally be greater than zero.
0x04	0–31	**Tx data buffer pointer**	The transmit buffer pointer, which always points to the first location of the associated data buffer, may be even or odd.The buffer may reside in either internal or external memory.

Data length (the second half word of a TxBD) is the number of octets the CP should send from this BD's data buffer. It is never modified by the CP.

Tx buffer pointer (the third and fourth half words of a TxBD) always points to the first location of the buffer in internal or external memory. The pointer may be even or odd.

35.6.3 USB Transmit Buffer Descriptor (TxBD) for Host

The TxBD described in this section is used when the packet-level interface is active. See Section 35.5.1.1, "Packet-Level Interface," for more information.

Data to be transmitted with the USB to the CP is arranged in buffers referenced by the TxBD ring. The first word of the TxBD contains status and control bits.

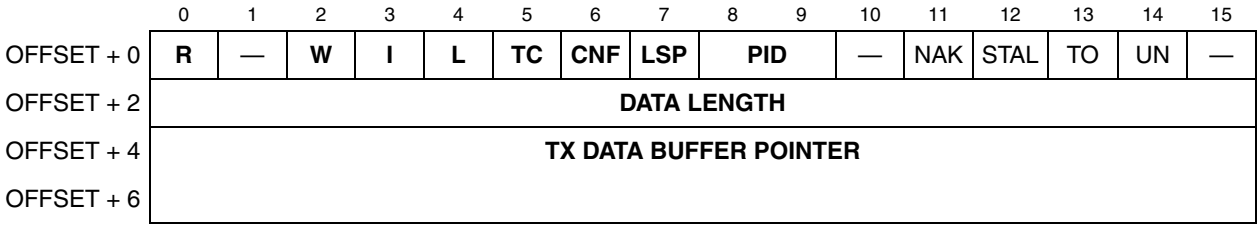


Figure 35-20. USB Transmit Buffer Descriptor (TxBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 35-17 describes USB TxBD fields.

Table 35-18. USB Host TxBD Fields

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASEx). The number of TxBDs in this table is programmable, and is determined only by the TxBD[W] and the overall space constraints of the dual-port RAM.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 The TXB or TXE bit in the event register is set when this buffer is serviced. TXB and TXE can cause interrupts if they are enabled.
	4	**L**	Last 0 Buffer does not contain the last byte of the message 1 Buffer contains the last byte of the message
	5	TC	Transmit CRC. Valid only when the L bit is set; otherwise it is ignored. Prepare TC before sending data. 0 Transmit end-of-packet after the last data byte. This setting can be used for testing purposes to send a bad CRC after the data. 1 Transmit the CRC sequence after the last data byte.

Table 35-18. USB Host TxBD Fields (continued)

Offset	Bits	Name	Description
	6	CNF	Transmit confirmation. Valid only when the L bit is set; otherwise it is ignored. Applies to multi-frame enabled endpoints (USEP*n*[MF] = 1); see Section 35.5.7.3, "USB Endpoint Registers (USEP1–USEP4)." 0 Continue to load the transmit FIFO with the next packet. No handshake or response is expected from the function for this packet. 1 Wait for handshake or response from the function before starting the next packet, or this is the last packet. Do not clear CNF for a token preceding a data packet unless the data packet's BD is ready.
	7	LSP	Low-speed transaction. Use for tokens only. 0 The following transaction is with the host or a full-speed device. 1 The following transaction is with a low-speed device. Required only for tokens. Note that LSP should always be cleared in slave mode.
	8–9	**PID**	Packet ID. This bit field is valid for the first BD of a packet; otherwise it is ignored. 0X Do not append PID to the data. 10 Transmit DATA0 PID before sending the data. 11 Transmit DATA1 PID before sending the data.
	10	—	Reserved, should be cleared.
	11	NAK[1]	NAK received. Indicates that the endpoint has responded with a NAK handshake. The packet was received error-free; however, the endpoint could not accept it.
	12	STAL[1]	STALL received. Indicates that the endpoint has responded with a STALL handshake. The endpoint needs attention through the control pipe.
	13	TO[1]	Time out. Indicates that the endpoint failed to acknowledge the packet.
	14	UN[1]	Underrun. Indicates that the USB encountered a transmitter underrun condition while sending the buffer.
	15	—	Reserved, should be cleared.
0x02	0–15	**Data length**	The data length is the number of octets that the CP should transmit from this BD's data buffer. It is never modified by the CP. This value should normally be greater than zero.
0x04	0–31	**Tx data buffer pointer**	The transmit buffer pointer, which always points to the first location of the associated data buffer, may be even or odd.The buffer may reside in either internal or external memory.

[1] Written by the USB controller after it finishes sending the associated data buffer.

Data length (the second half word of a TxBD) is the number of octets the CP should send from this BD data buffer. It is never modified by the CP.

Tx buffer pointer (the third and fourth half words of a TxBD) always points to the first location of the buffer in internal or external memory. The pointer may be even or odd.

35.6.4 USB Transaction Buffer Descriptor (TrBD) for Host

The TrBD described in this section is used when the transaction-level interface is active. See Section 35.5.1.2, "Transaction-Level Interface," for more information.

Data to be transmitted with the USB to the CP by is arranged in buffers referenced by the TrBD ring. The first word of the TrBD contains status and control bits.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0x0	**R**	—	**W**	**I**	**L**	**TC**	**CNF**	**LSP**	**PID**		RXER	NAK	STAL	TO	UN	BOV
OFFSET + 0x2	**DATA LENGTH**															
OFFSET + 0x4	**DATA BUFFER POINTER**															
OFFSET + 0x6																
OFFSET + 0x8	**TOK**		—	**ISO**	—	**ENDP**					**ADDR**					
OFFSET + 0xA	Reserved															

Figure 35-21. USB Transaction Buffer Descriptor (TrBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 35-19 describes USB TrBD fields.

Table 35-19. USB Host TrBD Fields

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TrBD table. 1 This is the last BD in the TrBD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASE). The number of TrBDs in this table is programmable, and is determined only by the TrBD[W] and the overall space constraints of the dual-port RAM.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 The TXB bit in the event register is set when this buffer is serviced. TXB can cause an interrupt if it is enabled.
	4	**L**	Last This bit should always be 1 since each TrBD represents an entire transaction.
	5	**TC**	Transmit CRC. Append CRC to transmitted data packet. 0 Transmit end-of-packet after the last data byte. This setting can be used for testing purposes to send a bad CRC after the data. 1 Transmit the CRC sequence after the last data byte.
	6	**CNF**	Transmit confirmation. This bit should always be set to 1 to obtain confirmation for each transaction.

Table 35-19. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
	7	**LSP**	Low-speed transaction. 0 This transaction is with the host or a full-speed device. 1 This transaction is with a low-speed device. Transmit a PRE packet before the token.
	8–9	**PID**	Packet ID. For OUT/SETUP transactions, this field is prepared by the user with the following values: 0X Do not append PID to the data packet. 10 Transmit DATA0 PID before sending the data packet. 11 Transmit DATA1 PID before sending the data packet. For IN transactions, this field is provided by the USB host controller with the following values: 00 Buffer contains DATA0 packet. 01 Buffer contains DATA1 packet.
	10	RXER[1]	Receive error. This bit indicates that an error was detected while receiving the data packet of an IN transaction. If RXER is 1, bits 11-15 have a different meaning as explained below.
	11	NAK/NO[1]	RXER = 0: NAK received. Indicates that the endpoint has responded with a NAK handshake (OUT transaction). The packet was received error-free; however, the endpoint could not accept it. RXER = 1: Rx non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data has been placed into the associated data buffer.
	12	STAL/AB[1]	RXER = 0: STALL received. Indicates that the endpoint has responded with a STALL handshake (OUT transaction). The endpoint needs attention through the control pipe. RXER = 1: Frame aborted. Bit stuff error occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	13	TO/CR[1]	RXER = 0: Time out. Indicates that the endpoint failed to acknowledge the token (IN transaction) or the data packet (OUT/SETUP transaction). RXER = 1: CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer. Written by the USB controller after the received data has been placed into the associated data buffer.
	14	UN/OV[1]	RXER = 0: Underrun. Indicates that the USB encountered a transmit FIFO underrun condition while sending the data packet (OUT/SETUP transaction). RXER = 1: Overrun. An internal receive FIFO overrun occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	15	BOV[1]	Buffer overflow. IN transactions only. Indicates that the number of received bytes is larger than the buffer size as provided in the Data Length field.
0x02	0–15	**Data Length**	For OUT/SETUP transactions, the user prepares this field with the number of bytes to be sent from the data buffer. It will not be modified by the CP. For IN transactions, the user prepares this field with the size of the data buffer, which must be divisible by 4. The CP will return the actual number of bytes written to the data buffer. If the number of received bytes, including the 2-byte CRC, is larger than the data buffer, the BOV bit will be set by the CP.
0x04	0–31	**Data Buffer Pointer**	The data buffer pointer. The buffer may reside in either internal or external memory. For OUT/SETUP transactions, this points to the buffer containing the data packet to transmit. It may have any alignment. For IN transactions, this points to the buffer into which the data packet should be received, The pointer must be divisible by 4.

Table 35-19. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
0x08	0–1	**TOK**	Token type This field determines the type of token to be transmitted and the type of transaction. 00 SETUP 01 OUT 10 IN 11 Reserved
	2	—	Reserved, should be cleared.
	3	**ISO**	Isochronous This bit indicates that the transaction is isochronous, so no handshake is required. 0 Bulk/control/interrupt. The handshake packet is automatically expected or generated by the USB host controller. 1 Isochronous. No handshake packets are expected or generated. This bit actually controls the value that is written to USEP1[TM] before processing this transaction.
	5	—	Reserved, should be cleared.
	5–8	**ENDP**	Endpoint This field indicates the endpoint number to be included in the token.
	9–15	**ADDR**	Address This field indicates the device address to be included in the token.
0x0A	0–15	—	Reserved, should be cleared.

[1] Written by the USB controller after it finishes sending or receiving the associated data buffer.

35.7 USB CP Commands

The following transmit commands are issued to the CP command register (CPCR). Refer to Section 21.3.1, "CP Command Register (CPCR)."

35.7.1 STOP Tx Command

This command disables the transmission of data on the selected endpoint. After issuing the command the corresponding endpoint FIFO should be flushed. No further transmissions will take place until the restart Tx command is issued.

35.7.2 RESTART Tx Command

This command enables the transmission of data from the corresponding endpoint on the USB. This command is expected by the USB controller after a STOP Tx command, or after transmission error (underrun or time-out).

35.8 USB Controller Errors

The USB controller reports frame reception and transmission error conditions using the BDs and the USB event register (USBER). Transmission errors are shown in Table 35-20. Errors which exist exclusively in host mode or function mode are marked as such.

Table 35-20. USB Controller Transmission Errors

Error	Description
Transmit underrun	If an underrun occurs, the transmitter forces a bit stuffing violation, terminates buffer transmission, closes the buffer, sets TxBD[UN] and the corresponding USBER[TXE*n*]. The endpoint resumes transmission after the RESTART TX ENDPOINT command is received.
Transmit timeout	Transmit packet not acknowledged. If a timeout occurs, the controller tries to retransmit if USEP*n*[RTE] = 1. If RTE = 0 or the second attempt fails, the controller closes the buffer and sets TxBD[TO] and USBER[TXE*n*]. The endpoint resumes transmission after receiving a RESTART TX ENDPOINT command.
Tx data not ready	For **USB function mode** only. This error occurs if an IN token is received, but the corresponding endpoint's transmit FIFO is empty, or if the target endpoint is configured to NAK or STALL. The controller sets USBER[TXE*n*].
Reception of NAK or STALL handshake	For **USB host mode** only. If this error occurs, the channel closes the buffer, sets the corresponding status bit in the TxBD (NAK or STAL), and sets the TXE bit in the USB event register. The host will resume transmission after reception of the RESTART TRANSMIT command.

Table 35-21 describes the USB controller reception errors.

Table 35-21. USB Controller Reception Errors

Error	Description
Overrun error	If the 16-byte receive FIFO overruns, the previously received byte is overwritten. The controller closes the buffer and sets both RxBD[OV] and USBER[RXB]. For **USB function mode** the NAK handshake is sent after the end of the received packet if the packet was received error-free.
Busy error	A frame was received and discarded due to lack of buffers. The controller sets USBER[BSY].
Non octet-aligned packet	If this error occurs, the controller writes the received data to the buffer, closes the buffer and sets both RxBD[NO] and USBER[RXB].
CRC error	When a CRC error occurs, the controller closes the buffer, and sets both RxBD[CR] and USBER[RXB]. In isochronous mode (USEP*n*[TM] = 0b11), the USB controller reports a CRC error; however, there are no handshake packets (ACK) and the transfer continues normally when an error occurs.
Buffer overflow	For **USB host mode** packet-level interface only. If the received data packet is larger than the allocated buffer, the remaining data is discarded, and TrBD[BOV] is set. The TXE1 interrupt bit is set.

35.9 USB Function Controller Initialization Example

The following is an example initialization sequence for the USB controller operating in function mode. It can be used to set up two function endpoints to fill transmit FIFOs so that data is ready for transmission when an IN token is received from the USB. The tokens can be generated using a USB traffic generator.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.

2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and USBOE.
3. Clear FRAME_N.
4. Write (DPRAM+0x500) to EP1PTR, and (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
5. Write 0xBC80_0004 to DPRAM+0x20 to set up the TxBD[Status and Control, Data Length] fields of endpoint 1.
6. Write DPRAM+0x200 to DPRAM+0x24 to set up the TxBD[Buffer Pointer] field of endpoint 1.
7. Write 0xBCC0_0004 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of endpoint 2.
8. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of endpoint 2.
9. Write 0xCAFE_CAFE to DPRAM+0x200 to set up the endpoint 1 Tx data pattern.
10. Write 0xFACE_FACE to DPRAM+0x210 to set up the endpoint 2 Tx data pattern.
11. Write 0x0000_0020 to DPRAM+0x500 to set up the RBASE and TBASE fields of the endpoint 1 parameter RAM.
12. Write 0x1818_0100 to DPRAM+0x504 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 1 parameter RAM.
13. Write 0x0000_0020 to DPRAM+0x508 to set up the RBPTR and TBPTR fields of the endpoint 1 parameter RAM.
14. Clear the TSTATE field of the endpoint 1 parameter RAM.
15. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
16. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
17. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
18. Clear the TSTATE field of the endpoint 2 parameter RAM.
19. Write 0x0000 to USEP1: Endpoint Number 0, control transfer, one packet only, and normal handshake.
20. Write 0x7200 to USEP2: Endpoint Number 7, bulk transfer, one packet only, and normal handshake.
21. Write 0x00 to the USMOD for full-speed 12 Mbps function endpoint mode and disable the USB.
22. Write 0x05 to the USAD for slave address 5.
23. Set USMOD[EN] to enable the USB controller.
24. Write 0x80 to USCOM to start filling the Tx FIFO with endpoint 1 data ready for transmission when an IN token is received.
25. Write 0x81 to USCOM to start filling the Tx FIFO with endpoint 2 data ready for transmission when an IN token is received.
26. Generate an IN token to address 5, endpoint number 0, control.
27. Generate an IN token to address 5, endpoint number 7, bulk.

35.10 Programming the USB Host Controller (Packet-Level)

The MPC8555E implementation of a USB host uses the endpoint represented by USEP1 to control the host transmission and reception. The other endpoints are typically not used, except for testing purposes (loopback).

Programming the USB controller to act as host is similar to configuring an endpoint for function operation. A general outline of how to program the host controller follows. (A more detailed example can be found in Section 35.10.1, "USB Host Controller Initialization Example.")

- Set the host bit in the mode register (USBMOD[HOST] = 1) to configure the controller as a host.
- Set the multi-frame bit in the endpoint configuration register (USEP1[MF] = 1) to allow SETUP/OUT tokens and DATA0/DATA1 packets to be sent back-to-back.
- Prepare tokens in separate BDs.
- Using software, append the CRC5 as part of the transmitted data because the CPM does not support automatic CRC5 generation.
- Clock the USB host controller as a high speed function (48-MHz reference clock).
- For low-speed transactions with an external hub, set TxBD[LSP] in the token's BD. This causes the USB host controller to generate a preamble (PRE token) at full speed before changing the transmit rate to low speed and sending the data packet. After completion of the transaction, the host returns to full-speed operation. Note that LSP should be set only for token BDs.

35.10.1 USB Host Controller Initialization Example

The following is a local loopback example initialization sequence for the USB controller operating as a host. It can be used to set up the host endpoint and one function endpoint to demonstrate an IN token transaction.

1. Program CMXSCR to provide a 48-MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Write (DPRAM+0x500) to EP1PTR, (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
4. Write 0x0000_0020 to DPRAM+0x500 to set up the RBASE and TBASE fields of the host endpoint parameter RAM.
5. Write 0x1818_0100 to DPRAM+0x504 to set up the RFCR, TFCR, and MRBLR fields of the host endpoint parameter RAM.
6. Write 0x0000_0020 to DPRAM+0x508 to set up the RBPTR and TBPTR fields of the host endpoint parameter RAM.
7. Clear the TSTATE field of the host endpoint parameter RAM.
8. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
9. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.

10. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
11. Clear the TSTATE field of the endpoint 2 parameter RAM.
12. Write 0xB000_0000 to DPRAM+0x00 to set up the RxBD[Status and Control, Data Length] fields of the host endpoint.
13. Write DPRAM+0x100 to DPRAM+0x04 to set up the RxBD[Buffer Pointer] field of the host endpoint.
14. Write 0xB800_0003 to DPRAM+0x20 to set up the TxBD[Status and Control, Data Length] fields of the host endpoint.
15. Write DPRAM+0x200 to DPRAM+0x24 to set up the TxBD[Buffer Pointer] field of the host endpoint.
16. Write 0xBC80_0003 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of the function endpoint.
17. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of the function endpoint.
18. Write 0x698560 to DPRAM+0x200 to set up the host endpoint Tx data pattern. This pattern consists of the IN token and the CRC5.
19. Write 0xABCD_1234 to DPRAM+0x210 to set up the function endpoint Tx data pattern.
20. Write 0x0020 to USEP1 for the host: non-isochronous transfer, multi-packet, packet-level interface.
21. Write 0x1100 to USEP2 for the function: interrupt transfer, one packet only.
22. Write 0x06 to USMOD for full-speed 12 Mbps signaling, local loopback configuration (test and host modes set), and disable the USB.
23. Write 0x05 to the USAD for slave address 5.
24. Set USMOD[EN] to enable the USB controller.
25. Write 0x81 to the USCOM to start filling the Tx FIFO with endpoint 2 data to be ready for transmission when an IN token is received.
26. Write 0x80 to the USCOM to start transmitting the IN token.

The expected results are as follows:

- TxBD[Status and Control] of the host endpoint should contain 0x3800.
- TxBD[Data Length] of the host endpoint should contain 0x0003.
- TxBD[Status and Control] of endpoint 2 should contain 0x3C80.
- TxBD[Data Length] of endpoint 2 should contain 0x0003.
- RxBD[Status and Control] of the host endpoint should contain 0x3C00.
- RxBD[Data Length] of the host endpoint should contain 0x0005.
- The receive buffer of the host endpoint should contain 0xABCD_122B, 0x42xx_xxxx.

35.11 Programming the USB Host Controller (Transaction-Level)

The MPC8555E implementation of a USB host uses the endpoint represented by USEP1 to control the host transmission and reception. The other endpoints are typically not used, except for testing purposes (loopback).

Programming the USB controller to act as host is similar to configuring an endpoint for function operation. A general outline of how to program the host controller follows. (A more detailed example can be found in Section 35.11.1, "USB Host Controller Initialization Example.")

- Set the host bit in the mode register (USBMOD[HOST] = 1) to configure the controller as a host.
- Set the multi-frame bit in the endpoint configuration register (USEP1[MF] = 1) to allow SETUP/OUT tokens and DATA0/DATA1 packets to be sent back-to-back.
- Set USEP1[RTE] to enable the transaction-level interface.
- Clock the USB host controller as a high speed function (48-MHz reference clock).
- For low-speed transactions with an external hub, set TrBD[LSP]. This causes the USB host controller to generate a preamble (PRE token) at full speed before changing the transmit rate to low speed and sending the token. After completion of the transaction, the host returns to full-speed operation.

35.11.1 USB Host Controller Initialization Example

The following is a local loopback example initialization sequence for the USB controller operating as a host. It can be used to set up the host endpoint and one function endpoint to demonstrate an IN token transaction.

1. Program CMXSCR to provide a 48-MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Write (DPRAM+0x500) to EP1PTR, (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
4. Write 0x0020 to DPRAM+0x502 to set up the TBASE field of the host endpoint parameter RAM.
5. Write 0x1818 to DPRAM+0x504 to set up the RFCR and TFCR fields of the host endpoint parameter RAM.
6. Write 0x0020 to DPRAM+0x50a to set up the TBPTR field of the host endpoint parameter RAM.
7. Clear the TSTATE field of the host endpoint parameter RAM.
8. Initialize the HIMMR field of the host endpoint parameter RAM.
9. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
10. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
11. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
12. Clear the TSTATE field of the endpoint 2 parameter RAM.

13. Write 0xB800_0040 to DPRAM+0x20 to set up the TrBD[Status and Control, Data Length] fields of the host endpoint.
14. Write DPRAM+0x100 to DPRAM+0x24 to set up the TrBD[Buffer Pointer] field of the host endpoint.
15. Write 0x8085 to DPRAM+0x28 to set up the TrBD token fields of the host endpoint.
16. Write 0xBC80_0003 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of the function endpoint.
17. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of the function endpoint.
18. Write 0xABCD_1234 to DPRAM+0x210 to set up the function endpoint Tx data pattern.
19. Write 0x0030 to USEP1 for the host: non-isochronous transfer, multi-packet, transaction-level interface.
20. Write 0x1100 to USEP2 for the function: interrupt transfer, one packet only.
21. Write 0x06 to USMOD for full-speed 12 Mbps signaling, local loopback configuration (test and host modes set), and disable the USB.
22. Write 0x05 to the USAD for slave address 5.
23. Set USMOD[EN] to enable the USB controller.
24. Write 0x81 to the USCOM to start filling the Tx FIFO with endpoint 2 data to be ready for transmission when an IN token is received.
25. Write 0x80 to the USCOM to start transmitting the IN token.

The expected results are as follows:

- TrBD[Status and Control] of the host endpoint should contain 0x3800.
- TrBD[Data Length] of the host endpoint should contain 0x0005.
- TxBD[Status and Control] of endpoint 2 should contain 0x3C80.
- TxBD[Data Length] of endpoint 2 should contain 0x0003.
- The receive buffer of the host endpoint should contain 0xABCD_122B, 0x42xx_xxxx.

Chapter 36
Serial Management Controllers (SMCs)

The two serial management controllers (SMCs) are full-duplex ports that can be configured independently to support one of three protocols or modes—UART, transparent, or general-circuit interface (GCI). Simple UART operation is used to provide a debug/monitor port in an application, which allows the SCCs to be free for other purposes. The SMC in UART mode is not as complex as that of the SCC in UART mode. The SMC clock can be derived from one of the internal baud rate generators or from an external clock signal. However, the clock should be a 16× clock.

In totally transparent mode, the SMC can be connected to a TDM channel (such as a T1 line) or directly to its own set of signals. The receive and transmit clocks are derived from the TDM channel, the internal baud rate generators, or from an external 1× clock. The transparent protocol allows the transmitter and receiver to use the external synchronization signal. The SMC in transparent mode is not as complex as that of the SCC in transparent mode.

Each SMC supports the C/I and monitor channels of the GCI bus, for which the SMC connects to a time-division multiplex (TDM) channel in a serial interface (SI*x*). SMCs support loopback and echo modes for testing. The SMC receiver and transmitter are double-buffered, corresponding to an effective FIFO size (latency) of two characters. Chapter 23, "Serial Interface with Time-Slot Assigner," describes GCI interface configuration.

Figure 36-1 shows the SMC block diagram.

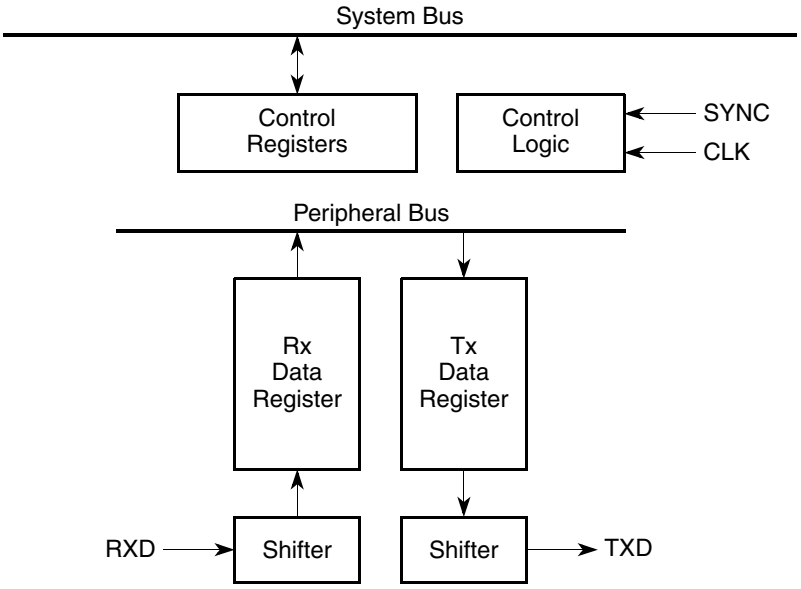


Figure 36-1. SMC Block Diagram

The receive data source can be L1RXD if the SMC is connected to a TDM channel of an SI*x*, or SMRXD if it is connected to the NMSI. The transmit data source can be L1TXD if the SMC is connected to a TDM or SMTXD if it is connected to the NMSI.

If the SMC is connected to a TDM, the SMC receive and transmit clocks can be independent from each other, as defined in Chapter 23, "Serial Interface with Time-Slot Assigner." However, if the SMC is connected to the NMSI, receive and transmit clocks must be connected to a single clock source (SMCLK), an internal signal name for a clock generated from the bank of clocks. SMCLK originates from an external signal or one of the four internal baud rate generators.

An SMC connected to a TDM derives a synchronization pulse from the TSA. An SMC connected to the NMSI using transparent protocol can use $\overline{\text{SMSYN}}$ for synchronization to determine when to start a transfer. $\overline{\text{SMSYN}}$ is not used when the SMC is in UART mode.

36.1 Features

The following is a list of the SMC's main features:

- Each SMC can implement the UART protocol on its own signals.
- Each SMC can implement a totally transparent protocol on a multiplexed or nonmultiplexed line. This mode can also be used for a fast connection between microprocessors.
- Each SMC channel fully supports the C/I and monitor channels of the GCI (IOM-2) in ISDN applications.
- Two SMCs support the two sets of C/I and monitor channels in the SCIT channels 0 and 1
- Full-duplex operation
- Local loopback and echo capability for testing

36.2 Common SMC Settings and Configurations

The following sections describe settings and configurations that are common to the SMCs.

36.2.1 SMC Mode Registers (SMCMR1, SMCMR2)

The SMC mode registers (SMCMR1 and SMCMR2), shown in Figure 36-2, select the SMC mode as well as mode-specific parameters. The functions of SMCMR[8–15] are the same for each protocol. Bits 0–7 vary according to protocol selected by the SM bits.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field: UART	—	CLEN				SL	PEN	PM	—		SM		DM		TEN	REN
Transparent						—	BS	REVD								
GCI						ME	—	C#								
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x0x9_1A82(SMCMR1), 0x0x9_1A92(SMCMR2)															

Figure 36-2. SMC Mode Registers (SMCMR1, SMCMR2)

Table 36-1 describes SMCMR fields.

Table 36-1. SMCMR1, SMCMR2 Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1–4	CLEN	Character length (UART). Number of bits in the character minus one. The total is the sum of 1 (start bit always present) + number of data bits (5–14) + number of parity bits (0 or 1) + number of stop bits (1 or 2). For example, for 8 data bits, no parity, and 1 stop bit, the total number of bits in the character is 1 + 8 + 0 + 1 = 10. So, CLEN should be programmed to 9. Characters range from 5–14 bits. If the data bit length is less than 8, the msbs of each byte in memory are not used on transmit and are written with zeros on receive. If the length is more than 8, the msbs of each 16-bit word are not used on transmit and are written with zeros on receive. The character must not exceed 16 bits. For a 14-bit data length, set SL to one stop bit and disable parity. For a 13-bit data length with parity enabled, set SL to one stop bit. Writing values 0 to 3 to CLEN causes erratic behavior.
		Character length (transparent). The values 3–15 specify 4–16 bits per character. If a character is less than 8 bits, the most-significant bits of the byte in buffer memory are not used on transmit and are written with zeros on receive. If character length is more than 8 bits but less than 16, the most-significant bits of the half-word in buffer memory are not used on transmit and are written with zeros on receive. Note: Using values 0–2 causes erratic behavior. Larger character lengths increase an SMC channel's potential performance and lowers the performance impact of other channels. For instance, using 16- rather than 8-bit characters is encouraged if 16-bit characters are acceptable in the end application.
		Character length (GCI). Number of bits in the C/I and monitor channels of the SCIT channels 0 or 1. (Values 0–15 correspond to 1–16 bits.) CLEN should be 13 for SCIT channel 0 or GCI (8 data bits, plus A and E bits, plus 4 C/I bits = 14 bits). It should be 15 for the SCIT channel 1 (8 data, bits, plus A and E bits, plus 6 C/I bits = 16 bits).
5	SL	Stop length (UART) 0 One stop bit 1 Two stop bits
	—	Reserved, should be cleared. (transparent)
	ME	Monitor enable (GCI) 0 The SMC does not support the monitor channel. 1 The SMC supports the monitor channel.

Table 36-1. SMCMR1, SMCMR2 Field Descriptions (continued)

Bits	Name	Description
6	PEN	Parity enable (UART) 0 No parity 1 Parity is enabled for the transmitter and receiver as determined by the PM bit.
	BS	Byte sequence(transparent). Controls the byte transmission sequence if REVD is set for a character length greater than 8 bits. Clear BS to maintain behavior compatibility with MC68360 QUICC. 0 Normal mode. This should be selected if the character length is not larger than 8 bits. 1 Transmit lower address byte first.
	—	Reserved, should be cleared. (GCI)
7	PM	Parity mode (UART) 0 Odd parity 1 Even parity
	REVD	Reverse data (transparent) 0 Normal mode 1 Reverse the character bit order. The msb is sent first.
	C#	SCIT channel number (GCI) 0 SCIT channel 0 1 SCIT channel 1. Required for Siemens ARCOFI and SGS S/T chips
8–9	—	Reserved, should be cleared.
10–11	SM	SMC mode 00 GCI or SCIT support 01 Reserved. 10 UART (must be selected for SMC UART operation) 11 Totally transparent operation
12–13	DM	Diagnostic mode. 00 Normal operation 01 Local loopback mode 10 Echo mode 11 Reserved
14	TEN	SMC transmit enable 0 SMC transmitter disabled 1 SMC transmitter enabled
15	REN	SMC receive enable 0 SMC receiver disabled 1 SMC receiver enabled

36.2.2 SMC Buffer Descriptor Operation

In UART and transparent modes, the SMC's memory structure is like the SCC's, except that SMC-associated data is stored in buffers. Each buffer is referenced by a BD and organized in a BD table located in the dual-port RAM. See Figure 36-3.

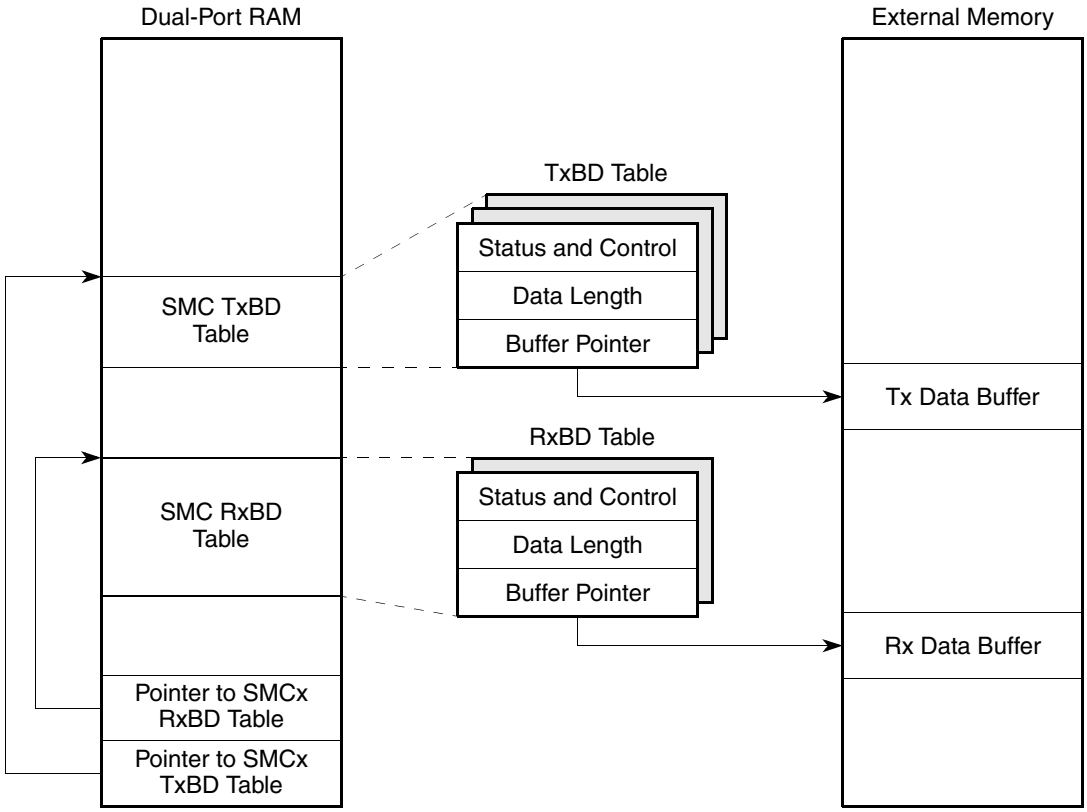


Figure 36-3. SMC Memory Structure

The BD table allows buffers to be defined for transmission and reception. Each table forms a circular queue. The CP uses BDs to confirm reception and transmission so that the processor knows buffers have been serviced. The data resides in external or internal buffers.

When SMCs are configured to operate in GCI mode, their memory structure is predefined to be one half-word long for transmit and one half-word long for receive. For more information on these half-word structures, see Section 36.5, “SMC in GCI Mode.”

36.2.3 SMC Parameter RAM

The CP accesses each SMC’s parameter table using a user-programmed pointer (SMC*x*_BASE) located in the parameter RAM; see Section 21.4.2, “Parameter RAM.” Each SMC parameter RAM table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11, and 12). The protocol-specific portions of the SMC parameter RAM are discussed in the sections that follow. The SMC parameter RAM shared by the UART and transparent protocols is shown in Table 36-2. Parameter RAM for GCI protocol is described in Table 36-16.

Table 36-2. SMC UART and Transparent Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	RxBDs and TxBDs base address. (BD table pointer) Define starting points in the dual-port RAM of the set of BDs for the SMC send and receive functions. They allow flexible partitioning of the BDs. By selecting RBASE and TBASE entries for all SMCs and by setting W in the last BD in each list, BDs are allocated for the send and receive side of every SMC. Initialize these entries before enabling the corresponding channel. Configuring BD tables of two enabled SMCs to overlap causes erratic operation. RBASE and TBASE should be a multiple of eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code. See Section 36.2.3.1, “SMC Function Code Registers (RFCR, TFCR).”
0x05	**TFCR**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. The most bytes the MPC8555E writes to a receive buffer before moving to the next buffer. It can write fewer bytes than MRBLR if a condition like an error or end-of-frame occurs, but it cannot exceed MRBLR. MPC8555E buffers should not be smaller than MRBLR. SMC transmit buffers are unaffected by MRBLR. Transmit buffers can be individually given varying lengths through the data length field. MRBLR can be changed while an SMC is operating only if it is done in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). This occurs when the CP shifts control to the next RxBD, so the change does not take effect immediately. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the SMC receiver is disabled. MRBLR should be greater than zero and should be even if character length exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state. [2] Can be used only by the CP.
0x0C	—	Word	Rx internal data pointer. [2] Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the next BD for each SMC channel that the receiver transfers data to when it is in idle state, or to the current BD during frame processing. After a reset or when the end of the BD table is reached, the CP initializes RBPTR to the value in RBASE. Most applications never need to write RBPTR, but it can be written when the receiver is disabled or when no receive buffer is in use.
0x12	—	Hword	Rx internal byte count. [2] A down-count value initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	—	Word	Rx temp [2] Can be used only by the CP.
0x18	TSTATE	Word	Tx internal state. [2] Can be used only by the CP.
0x1C	—	Word	Tx internal data pointer. [2] Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the next BD for each SMC channel the transmitter transfers data from when it is in idle state or to the current BD during frame transmission. After reset or when the end of the table is reached, the CP initializes TBPTR to the TBASE value. Most applications never need to write TBPTR, but it can be written when the transmitter is disabled or when no transmit buffer is in use. For instance, after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission.
0x22	—	Hword	Tx internal byte count. [2] A down-count value initialized with the TxBD data length and decremented with every byte the SDMA channels read.
0x24	—	Word	Tx temp. [2] Can be used only by the CP.

Table 36-2. SMC UART and Transparent Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x28	**MAX_IDL**	Hword	Maximum idle characters. (UART protocol-specific parameter) When a character is received on the line, the SMC starts counting idle characters received. If MAX_IDL idle characters arrive before the next character, an idle time-out occurs and the buffer closes, which sends an interrupt request to the core to receive data from the buffer. MAX_IDL demarcates frames in UART mode. Clearing MAX_IDL disables the function so the buffer never closes, regardless of how many idle characters are received. An idle character is calculated as follows: 1 + data length (5 to 14) + 1 (if parity bit is used) + number of stop bits (1 or 2). For example, for 8 data bits, no parity, and 1 stop bit, character length is 10 bits.
0x2A	IDLC	Hword	Temporary idle counter. (UART protocol-specific parameter) Down-counter in which the CP stores the current idle counter value in the MAX_IDL time-out process.
0x2C	BRKLN	Hword	Last received break length. (UART protocol-specific parameter) Holds the length of the last received break character sequence measured in character units. For example, if the receive signal is low for 20 bit times and the defined character length is 10 bits, BRKLN = 0x002, indicating that the break sequence is at least 2 characters long. BRKLN is accurate to within one character length.
0x2E	**BRKEC**	Hword	Receive break condition counter. (UART protocol-specific parameter) Counts break conditions on the line. A break condition may last for hundreds of bit times, yet BRKEC increments only once during that period.
0x30	**BRKCR**	Hword	Break count register (transmit). (UART protocol-specific parameter) Determines the number of break characters the UART controller sends. Set when the SMC sends a break character sequence after a STOP TRANSMIT command. For 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character is 10 zeros.
0x32	R_MASK	Hword	Temporary bit mask. (UART protocol-specific parameter)
0x34	—	Word	SDMA temp

[1] From the pointer value programmed in SMC*x*_BASE: SMC1_BASE at 0x87FC, SMC2_BASE at IMMR + 0x88FC.

[2] Not accessed for normal operation. May hold helpful information for experienced users and for debugging.

To extract data from a partially full receive buffer, issue a CLOSE RXBD command.

Certain parameter RAM values must be initialized before the SMC is enabled. Other values are initialized or written by the CP. Once values are initialized, software typically does not need to update them because activity centers mostly around transmit and receive BDs rather than parameter RAM. However, note the following:

- Parameter RAM can be read at any time.
- Values that pertain to the SMC transmitter can be written only if SMCMR[TEN] is zero or between the STOP TRANSMIT and RESTART TRANSMIT commands.
- Values for the SMC receiver can be written only when SMCMR[REN] is zero, or, if the receiver is previously enabled, after an ENTER HUNT MODE command is issued but before the CLOSE RXBD command is issued and REN is set.

36.2.3.1 SMC Function Code Registers (RFCR, TFCR)

The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 36-4 shows the register format.

	0	1	2	3	4	5	6	7
Field			GBL	BO		TC2	DTB	—
R/W	R/W							
Offset	SMC base + 0x04 (RFCR)/SMC base + 0x05 (TFCR)							

Figure 36-4. SMC Function Code Registers (RFCR, TFCR)

Table 36-3 describes FCR fields.

Table 36-3. RFCR, TFCR Field Descriptions

Bit	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global access bit 0 Disable memory snooping 1 Enable memory snooping
3–4	**BO**	Byte ordering. Selects byte ordering of the data buffer. 00 The DEC/Intel convention (swapped operation or little-endian). The transmission order of bytes within a buffer word is opposite of big-endian mode. (32-bit port size memory only). 01 Munged little-endian. As data is sent onto the serial line from the buffer, the LSB of the buffer double word contains data to be sent earlier than the MSB of the same double word. 1x Big-endian byte ordering (normal operation). As data is sent onto the serial line from the buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same word.
5	**TC2**	Transfer code 2. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Data bus indicator 0 Use system bus for SDMA operation 1 Reserved.
7	—	Reserved, should be cleared.

36.2.4 Disabling SMCs On-The-Fly

An SMC can be disabled and re-enabled later by ensuring that buffers are closed properly and new data is transferred to or from a new buffer. Such a sequence is required if the parameters to be changed are not dynamic. If the register or bit description states that dynamic changes are allowed, the sequences need not be followed and the register or bits may be changed immediately.

NOTE

The SMC does not have to be fully disabled for parameter RAM to be modified. Table 36-2 describes when parameter RAM values can be modified. To disable all SCCs, SMCs, the SPI, and the I^2C, use the CPCR to reset the CPM with a single command.

36.2.4.1 SMC Transmitter Full Sequence

Follow these steps to fully enable or disable the SMC transmitter:

1. If the SMC is sending data, issue a STOP TRANSMIT command to stop transmission smoothly. If the SMC is not sending, if TBPTR is overwritten, or if an INIT TX PARAMETERS command is executed, this command is not required.
2. Clear SMCMR[TEN] to disable the SMC transmitter and put it in reset state.
3. Update SMC transmit parameters, including the parameter RAM. To switch protocols or reinitialize parameters, issue an INIT TX PARAMETERS command.
4. Issue a RESTART TRANSMIT if an INIT TX PARAMETERS was issued in step 3.
5. Set SMCMR[TEN]. Transmission now begins using the TxBD that the TBPTR value pointed to as soon as the R bit is set in the TxBD.

36.2.4.2 SMC Transmitter Shortcut Sequence

This shorter sequence reinitializes transmit parameters to the state they had after reset.

1. Clear SMCMR[TEN].
2. Issue an INIT TX PARAMETERS command and make any additional changes.
3. Set SMCMR[TEN].

36.2.4.3 SMC Receiver Full Sequence

Follow these steps to fully enable or disable the receiver:

1. Clear SMCMR[REN]. Reception is aborted immediately, which disables the SMC receiver and puts it in a reset state.
2. Modify SMC receive parameters, including parameter RAM. To switch protocols or reinitialize SMC receive parameters, issue an INIT RX PARAMETERS command.
3. Issue a CLOSE RXBD command if INIT RX PARAMETERS was not issued in step 2.
4. Set SMCMR[REN]. Reception immediately uses the RxBD that RBPTR pointed to if E is set in the RxBD.

36.2.4.4 SMC Receiver Shortcut Sequence

This shorter sequence reinitializes receive parameters to their state after reset.

1. Clear SMCMR[REN].
2. Issue an INIT RX PARAMETERS command and make any additional changes.
3. Set SMCMR[REN].

36.2.4.5 Switching Protocols

To switch the protocol that the SMC is executing without resetting the board or affecting any other SMC, use one command and follow these steps:

1. Clear SMCMR[REN] and SMCMR[TEN].

2. Issue an INIT TX AND RX PARAMETERS COMMAND to initialize transmit and receive parameters. Make any additional SMCMR changes.
3. Set SMCMR[REN, TEN]. The SMC is now enabled with the new protocol.

36.2.5 Saving Power

When SMCMR[TEN, REN] are cleared, the SMC consumes little power.

36.2.6 Handling Interrupts in the SMC

Follow these steps to handle an interrupt in the SMC:

1. Once an interrupt occurs, read SMCE to identify the interrupt source. The SMCE bits are usually cleared at this time.
2. Process the TxBD to reuse it if SMCE[TXB] is set. Extract data from the RxBD if SMCE[RXB] is set. To send another buffer, set TxBD[R].
3. Execute the **rfi** instruction.

36.3 SMC in UART Mode

SMCs generally offer less functionality and performance in UART mode than do SCCs, which makes them more suitable for simpler debug/monitor ports instead of full-featured UARTs. SMCs do not support the following features in UART mode.

- $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ signals
- Receive and transmit sections clocked at different rates
- Fractional stop bits
- Built-in multidrop modes
- Freeze mode for implementing flow control
- Isochronous (1× clock) operation (A 16× clock is required for UART operation.)
- Interrupts on special control character reception
- Ability to transmit data on demand using the TODR
- SCCS register to determine idle status of the receive signal
- Other features for the SCCs as described in the GSMR

However, SMCs allow a data length of up to 14 bits; SCCs support up to 8 bits.

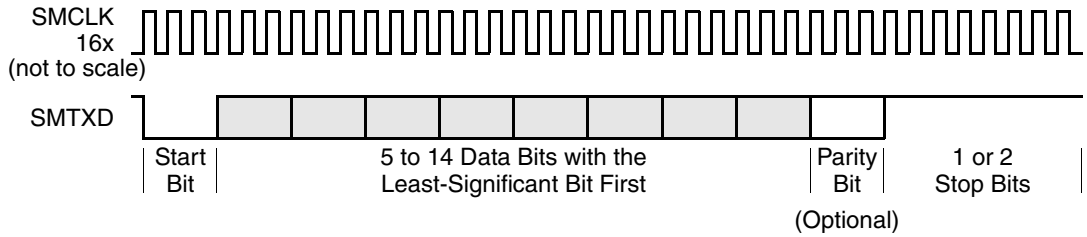


Figure 36-5. SMC UART Frame Format

36.3.1 Features

The following list summarizes the main features of the SMC in UART mode:

- Flexible message-oriented data structure
- Programmable data length (5–14 bits)
- Programmable 1 or 2 stop bits
- Even/odd/no parity generation and checking
- Frame error, break, and idle detection
- Transmit preamble and break sequences
- Received break character length indication
- Continuous receive and transmit modes

36.3.2 SMC UART Channel Transmission Process

The UART transmitter is designed to work with almost no intervention from the core. When the core enables the SMC transmitter, it starts sending idles. The SMC immediately polls the first BD in the transmit channel BD table and once every character time after that, depending on character length. When there is a message to transmit, the SMC fetches data from memory and starts sending the message.

When a BD data is completely written to the transmit FIFO, the SMC writes the message status bits into the BD and clears R. An interrupt is issued if the I bit in the BD is set. If the next TxBD is ready, the data from its buffer is appended to the previous data and sent over the transmit signal without any gaps between buffers. If the next TxBD is not ready, the SMC starts sending idles and waits for the next TxBD to be ready.

By appropriately setting the I bit in each BD, interrupts can be generated after each buffer, a specific buffer, or each block is sent. The SMC then proceeds to the next BD. If the CM bit is set in the TxBD, the R bit is not cleared, allowing a buffer to be automatically resent next time the CP accesses this buffer. For instance, if a single TxBD is initialized with the CM and W bits set, the buffer is sent continuously until R is cleared in the BD.

36.3.3 SMC UART Channel Reception Process

When the core enables the SMC receiver, it enters hunt mode and waits for the first character. The CP then checks the first RxBD to see if it is empty and starts storing characters in the buffer. When the buffer is full or the MAX_IDL timer expires (if enabled), the SMC clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the buffer's length, the SMC fetches the next BD, and, if it is empty, continues transferring data to this BD's buffer. If CM is set in the RxBD, the E bit is not cleared, so the CP can overwrite this buffer on its next access.

36.3.4 Programming the SMC UART Controller

UART mode is selected by setting SMCMR[SM] to 0b10. See Section 36.2.1, "SMC Mode Registers (SMCMR1, SMCMR2)." UART mode uses the same data structure as other modes. This structure supports multibuffer operation and allows break and preamble sequences to be sent. Overrun, parity, and

framing errors are reported through the BDs. At its simplest, the SMC UART controller functions in a character-oriented environment, whereas each character is sent with the selected stop bits and parity. They are received into separate 1-byte buffers. A maskable interrupt can be generated when each buffer is received.

Many applications can take advantage of the message-oriented capabilities that the SMC UART supports through linked buffers for sending or receiving. Data is handled in a message-oriented environment, so entire messages can be handled instead of individual characters. A message can span several linked buffers; each one can be sent and received as a linked list of buffers without core intervention, which simplifies programming and saves processor overhead. In a message-oriented environment, an idle sequence is used as the message delimiter. The transmitter can generate an idle sequence before starting a new message and the receiver can close a buffer when an idle sequence is found.

36.3.5 SMC UART Transmit and Receive Commands

Table 36-4 describes transmit commands issued to the CPCR.

Table 36-4. Transmit Commands

Command	Description
STOP TRANSMIT	Disables transmission of characters on the transmit channel. If the SMC UART controller receives this command while sending a message, it stops sending. The SMC UART controller finishes sending any data that has already been sent to its FIFO and shift register and then stops sending data. The TBPTR is not advanced when this command is issued. The SMC UART controller sends a programmable number of break sequences and then sends idles. The number of break sequences, which can be zero, should be written to the BRKCR before this command is issued to the SMC UART controller.
RESTART TRANSMIT	Enables characters to be sent on the transmit channel. The SMC UART controller expects it after disabling the channel in its SMCMR and after issuing the STOP TRANSMIT command. The SMC UART controller resumes transmission from the current TBPTR in the channel's TxBD table.
INIT TX PARAMETERS	Initializes transmit parameters in this serial channel's parameter RAM to their reset state and should only be issued when the transmitter is disabled. The INIT TX and RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Table 36-5 describes receive commands issued to the CPCR.

Table 36-5. Receive Commands

Command	Description
ENTER HUNT MODE	Use the CLOSE RXBD command instead ENTER HUNT MODE for an SMC UART channel.
CLOSE RXBD	Forces the SMC to close the current receive BD if it is currently being used and to use the next BD in the list for any subsequently received data. If the SMC is not receiving data, no action is taken.
INIT RX PARAMETERS	Initializes receive parameters in this serial channel parameter RAM to reset state. Issue it only if the receiver is disabled. INIT TX AND RX PARAMETERS resets both receive and transmit parameters.

36.3.6 Sending a Break

A break is an all-zeros character without stop bits. It is sent by issuing a STOP TRANSMIT command. After sending any outstanding data, the SMC sends a character of consecutive zeros, the number of which is the

sum of the character length, plus the number of start, parity, and stop bits. The SMC sends a programmable number of break characters according to BRKCR and then reverts to idle or sends data if a RESTART TRANSMIT is issued before completion. When the break completes, the transmitter sends at least one idle character before sending any data to guarantee recognition of a valid start bit.

36.3.7 Sending a Preamble

A preamble sequence provides a way to ensure that the line is idle before a new message transfer begins. The length of the preamble sequence is constructed of consecutive ones that are one-character long. If the preamble bit in a BD is set, the SMC sends a preamble sequence before sending that buffer. For 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of 10 ones would be sent before the first character in the buffer. If no preamble sequence is sent, data from two ready transmit buffers can be sent on the transmit signal with no delay between them.

36.3.8 Handling Errors in the SMC UART Controller

The SMC UART controller reports character reception error conditions through the channel BDs and the SMCE. Table 36-6 describes the reception errors. The SMC UART controller has no transmission errors.

Table 36-6. SMC UART Errors

Error	Description
Overrun	The SMC maintains a two-character length FIFO for receiving data. Data is moved to the buffer after the first character is received into the FIFO; if a receiver FIFO overrun occurs, the channel writes the received character into the internal FIFO. It then writes the character to the buffer, closes it, sets the OV bit in the BD, and generates the RXB interrupt if it is enabled. Reception then resumes as normal. Overrun errors that occasionally occur when the line is idle can be ignored.
Parity	The channel writes the received character to the buffer, closes it, sets the PR bit in the BD, and generates the RXB interrupt if it is enabled. Reception then resumes as normal.
Idle Sequence Receive	An idle is found when a character of all ones is received, at which point the channel counts consecutive idle characters. If the count reaches MAX_IDL, the buffer is closed and an RXB interrupt is generated. If no receive buffer is open, this does not generate an interrupt or any status information. The idle counter is reset each time a character is received.
Framing	The SMC received a character with no stop bit. When it occurs, the channel writes the received character to the buffer, closes the buffer, sets FR in the BD, and generates the RXB interrupt if it is enabled. When this error occurs, parity is not checked for the character.
Break Sequence	The SMC receiver received an all-zero character with a framing error. The channel increments BRKEC, generates a maskable BRK interrupt in SMCE, measures the length of the break sequence, and stores this value in BRKLN. If the channel was processing a buffer when the break was received, the buffer is closed with the BR bit in the RxBD set. The RXB interrupt is generated if it is enabled.

36.3.9 SMC UART RxBD

Using the BDs, the CP reports information about the received data on a per-buffer basis. Then it closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- An error is received during message processing
- A full receive buffer is detected
- A programmable number of consecutive idle characters are received

Figure 36-6 shows the format of the SMC UART RxBD.

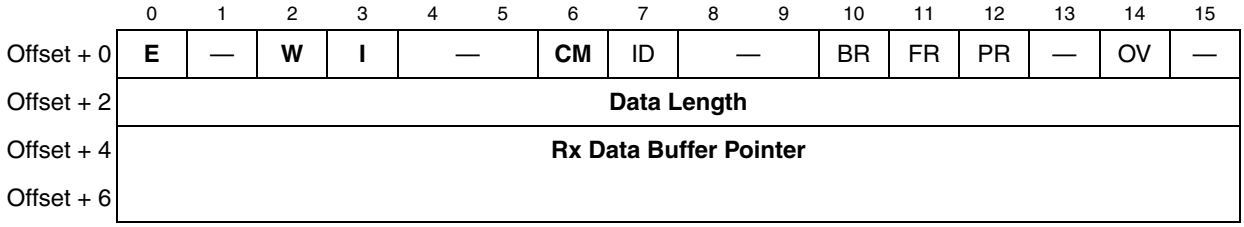


Figure 36-6. SMC UART RxBD

Table 36-7 describes RxBD fields.

Table 36-7. SMC UART RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The buffer is full or data reception stopped due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E is zero. 1 The buffer is empty or reception is in progress. This RxBD and its buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in RxBD table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is filled. 1 The SMCE[RXB] is set when this buffer is completely filled by the CP, indicating the need for the core to process the buffer. RXB can cause an interrupt if it is enabled.
4–5	—	Reserved, should be cleared.
6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear the E bit after this BD is closed, allowing the CP to automatically overwrite the buffer when it next accesses the BD. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7	ID	Buffer closed on reception of idles. Set when the buffer has closed because a programmable number of consecutive idle sequences is received. The CP writes ID after received data is in the buffer.

Table 36-7. SMC UART RxBD Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved, should be cleared.
10	BR	Buffer closed on reception of break. Set when the buffer closes because a break sequence was received. The CP writes BR after the received data is in the buffer.
11	FR	Framing error. Set when a character with a framing error is received and located in the last byte of this buffer. A framing error is a character with no stop bit. A new receive buffer is used to receive additional data. The CP writes FR after the received data is in the buffer.
12	PR	Parity error. Set when a character with a parity error is received in the last byte of the buffer. A new buffer is used for additional data. The CP writes PR after received data is in the buffer.
13	—	Reserved, should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception. The CP writes OV after the received data is in the buffer.
15	—	Reserved, should be cleared.

Data length represents the number of octets the CP writes into the buffer. After data is received in buffer, the CP only writes them once as the BD closes. Note that the memory allocated for this buffer should be no smaller than MRBLR. The Rx data buffer pointer points to the first location of the buffer and must be even. The buffer can be in internal or external memory. Figure 36-7 shows the UART RxBD process, showing RxBDs after they receive 10 characters, an idle period, and five characters (one with a framing error). The example assumes that MRBLR = 8.

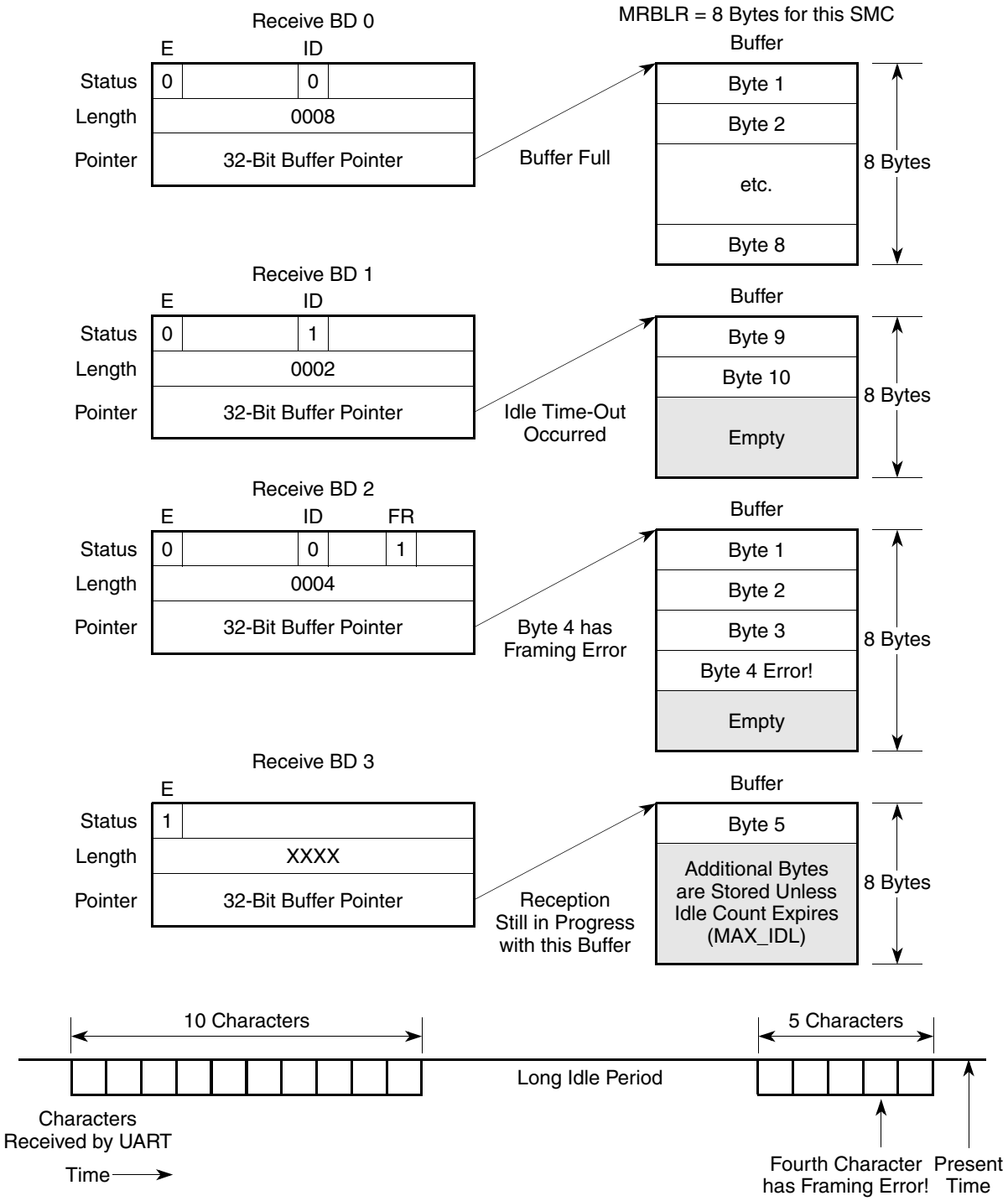


Figure 36-7. RxBD Example

36.3.10 SMC UART TxBD

Data is sent to the CP for transmission on an SMC channel by arranging it in buffers referenced by the channel TxBD table. Using the BDs, the CP confirms transmission or indicates error conditions so that the processor knows the buffers have been serviced. An SMC UART TxBD is displayed in Figure 36-8.

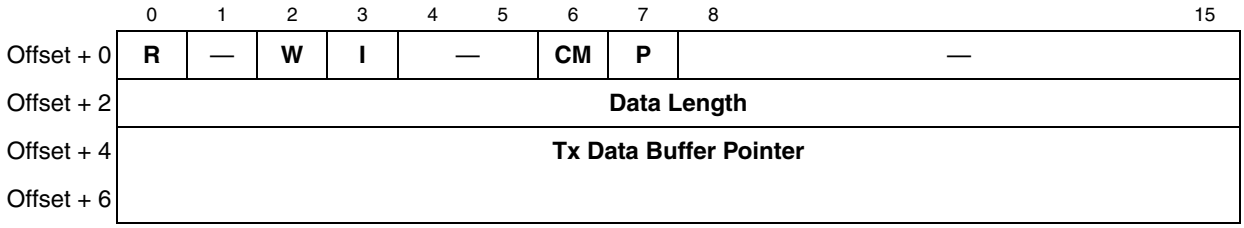


Figure 36-8. SMC UART TxBD

Table 36-8 describes SMC UART TxBD fields.

Table 36-8. SMC UART TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready 0 The buffer is not ready for transmission; BD and its buffer can be altered. The CP clears R after the buffer has been sent or an error occurs. 1 The buffer has not been completely sent. This BD cannot updated while R is set.
1	—	Reserved, should be cleared.
2	**W**	Wrap (final BD in the TxBD table) 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 The SMCE[TXB] is set when this buffer is serviced. TXB can cause an interrupt if it is enabled.
4–5	—	Reserved, should be cleared.
6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear R after this BD is closed and automatically retransmits the buffer when it accesses this BD next.
7	**P**	Preamble 0 No preamble sequence is sent. 1 The UART sends one all-ones character before it sends the data so that the other end detects an idle line before the data is received. If this bit is set and the data length of this BD is zero, only a preamble is sent.
8–15	—	Reserved, should be cleared.

Data length represents the number of octets that the CP should transmit from this BD data buffer. However, it is never modified by the CP and normally is greater than zero. It can be zero if P is set and only a preamble is sent. If there are more than 8 bits in the UART character, data length should be even. For example, to transmit three UART characters of 8-bit data, 1 start, and 1 stop, initialize the data length field to 3. To send three UART characters of 9-bit data, 1 start, and 1 stop, the data length field should 6, because

the three 9-bit data fields occupy three half words in memory (the 9 least-significant bits of each half word).

Tx data buffer pointer points to the first location of the buffer. It can be even or odd, unless the number of data bits in the UART character is greater than 8 bits. Then the buffer pointer must be even. For instance, the pointer to 8-bit data, 1 start, and 1 stop characters can be even or odd, but the pointer to 9-bit data, 1 start, and 1 stop characters must be even. The buffer can reside in internal or external memory.

36.3.11 SMC UART Event Register (SMCE)/Mask Register (SMCM)

The SMC event register (SMCE) generates interrupts and report events recognized by the SMC UART channel. When an event is recognized, the SMC UART controller sets the corresponding SMCE bit. Bits are cleared by writing a 1; writing 0 has no effect. The SMC mask register (SMCM) has the same bit format as SMCE. Setting an SMCM bit enables, and clearing it disables, the corresponding interrupt. All unmasked bits must be cleared before the CP clears the internal interrupt request. Figure 36-9 represents the SMCE/SMCM registers.

	0	1	2	3	4	5	6	7
Field	—	BRKE	—	BRK	—	BSY	TXB	RXB
Reset	0							
R/W	R/W							
Offset	0x0x9_1A86(SMCE1); 0x0x9_1A96(SMCE2)/ 0x0x9_1A8A(SMCM1); 0x0x9_1A9A(SMCM2)							

Figure 36-9. SMC UART Event Register (SMCE)/Mask Register (SMCM)

Table 36-9 describes SMCE/SMCM fields.

Table 36-9. SMCE/SMCM Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1	BRKE	Break end. Set no sooner than after one idle bit is received after the break sequence.
2	—	Reserved, should be cleared.
3	BRK	Break character received. Set when a break character is received. If a very long break sequence occurs, this interrupt occurs only once after the first all-zeros character is received.
4	—	Reserved, should be cleared.
5	BSY	Busy condition. Set when a character is received and discarded due to a lack of buffers. Set no sooner than the middle of the last stop bit of the first receive character for which there is no available buffer. Reception resumes when an empty buffer is provided.
6	TXB	Tx buffer. Set when the transmit data of the last character in the buffer is written to the transmit FIFO. Wait two character times to ensure that data is completely sent over the transmit signal.
7	RXB	Rx buffer. Set when a buffer is received and its associated RxBD is closed. Set no sooner than the middle of the last stop bit of the last character that is written to the receive buffer.

Figure 36-10 shows an example of the timing of various events in the SMCE.

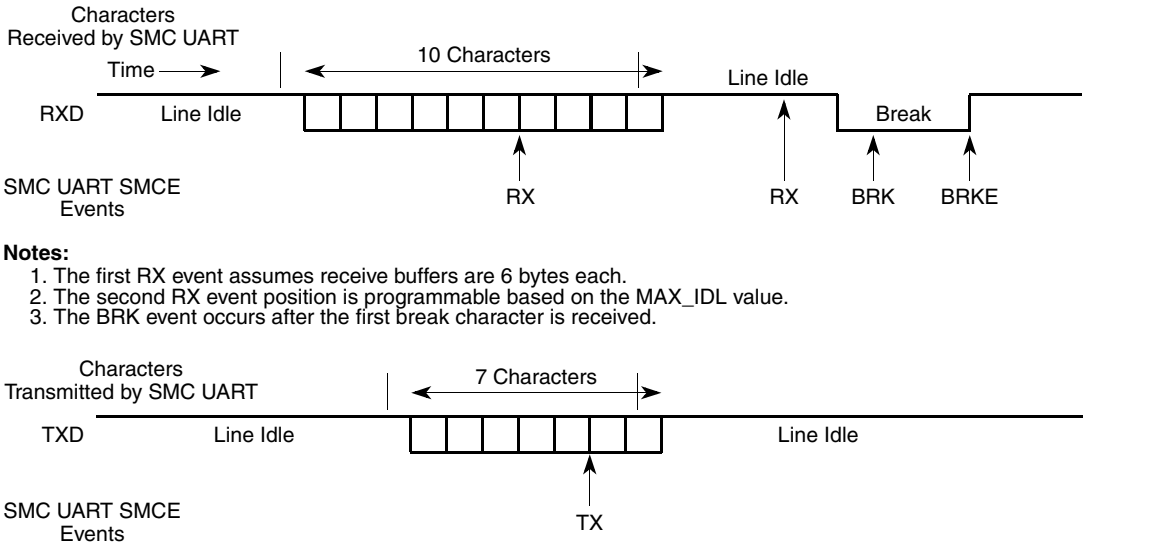


Figure 36-10. SMC UART Interrupts Example

36.3.12 SMC UART Controller Programming Example

The following initialization sequence assumes 9,600 baud, 8 data bits, no parity, and 1 stop bit in a 66-MHz system. BRG1 and SMC1 are used. (The SMC transparent programming example uses an external clock configuration; see Section 36.4.11, "SMC Transparent NMSI Programming Example.")

1. Configure the port D pins to enable SMTXD1 and SMRXD1. Set PPARD[8,9] and PDIRD[9]. Clear PDIRD[8] and PSORD[8,9].
2. Configure the BRG1. Write BRGC1 with 0x0001_035A. The DIV16 bit is not used and the divider is 429 (decimal). The resulting BRG1 clock is 16× the preferred bit rate.
3. Connect BRG1 to SMC1 using the CPM mux by clearing CMXSMR[SMC1, SMC1CS].
4. In address 0x87FC, assign a pointer to the SMC1 parameter RAM.
5. Assuming one RxBD at the beginning of dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
6. Write 0x1D01_0000 to CPCR to execute the INIT RX AND TX PARAMETERS command.
7. Write RFCR and TFCR with 0x10 for normal operation.
8. Write MRBLR with the maximum number of bytes per receive buffer. Assume 16 bytes, so MRBLR = 0x0010.
9. Write MAX_IDL with 0x0000 in the SMC UART-specific parameter RAM to disable the MAX_IDL functionality for this example.
10. Clear BRKLN and BRKEC in the SMC UART-specific parameter RAM.
11. Set BRKCR to 0x0001; if a STOP TRANSMIT COMMAND is issued, one break character is sent.

12. Initialize the RxBD. Assume the Rx data buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (not required), and 0x0000_1000 to RxBD[Buffer Pointer].
13. Assuming the Tx data buffer is at 0x0000_2000 in main memory and contains five 8-bit characters, write 0xB000 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
14. Write 0xFF to the SMCE1 register to clear any previous events.
15. Write 0x57 to the SMCM1 register to enable all possible SMC1 interrupts.
16. Write 0x0000_1000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Write 0xFFFF_FFFF to the SIU interrupt pending register low (SIPNR_L) to clear events.
17. Write 0x4820 to SMCMR to configure normal operation (not loopback), 8-bit characters, no parity, 1 stop bit. The transmitter and receiver are not yet enabled.
18. Write 0x4823 to SMCMR to enable the SMC transmitter and receiver. This additional write ensures that the TEN and REN bits are enabled last.

After 5 bytes are sent, the TxBD is closed. The receive buffer closes after receiving 16 bytes. Subsequent data causes a busy (out-of-buffers) condition since only one RxBD is ready.

36.4 SMC in Transparent Mode

Compared to the SCC in transparent mode, the SMCs generally offer less functionality, which helps them provide simpler functions and slower speeds. Transparent mode is selected by programming SMCMR[SM] to 0b10. Section 36.2.1, “SMC Mode Registers (SMCMR1, SMCMR2),” describes other protocol-specific bits in the SMCMR. The SMC in transparent mode does not support the following features:

- Independent transmit and receive clocks, unless connected to a TDM channel of an SI*x*
- CRC generation and checking
- Full $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ signals (supports only one $\overline{\text{SMSYN}}$ signal)
- Ability to transmit data on demand using the TODR
- Receiver/transmitter in transparent mode while executing another protocol
- 4-, 8-, or 16-bit SYNC recognition
- Internal DPLL support

However, the SMC in transparent mode provides a data character length option of 4 to 16 bits, whereas the SCCs provide 8 or 32 bits, depending on GSMR[RFW]. The SMC in transparent mode is also referred to as the SMC transparent controller.

36.4.1 Features

The following list summarizes the features of the SMC in transparent mode:

- Flexible data buffers
- Connects to a TDM bus using the TSA in an SI*x*

- Transmits and receives transparently on its own set of signals using a sync signal to synchronize the beginning of transmission and reception to an external event
- Programmable character length (4–16)
- Reverse data mode
- Continuous transmission and reception modes
- Four commands

36.4.2 SMC Transparent Channel Transmission Process

The transparent transmitter is designed to work with almost no core intervention. When the core enables the SMC transmitter in transparent mode, it starts sending idles. The SMC immediately polls the first BD in the transmit channel BD table and once every character time, depending on the character length (every 4 to 16 serial clocks). When there is a message to transmit, the SMC fetches the data from memory and starts sending the message when synchronization is achieved.

Synchronization can be achieved in two ways. First, when the transmitter is connected to a TDM channel, it can be synchronized to a time slot. Once the frame sync is received, the transmitter waits for the first bit of its time slot before it starts transmitting. Data is sent only during the time slots defined by the TSA. Secondly, when working with its own set of signals, the transmitter starts sending when $\overline{\text{SMSYN}}x$ is asserted.

When a BD data is completely written to the transmit FIFO, the L bit is checked and if it is set, the SMC writes the message status bits into the BD and clears the R bit. It then starts transmitting idles. When the end of the current BD is reached and the L bit is not set, only R is cleared. In both cases, an interrupt is issued according to the I bit in the BD. By appropriately setting the I bit in each BD, interrupts can be generated after each buffer, a specific buffer, or each block is sent. The SMC then proceeds to the next BD. If no additional buffers have been presented to the SMC for transmission and the L bit was cleared, an underrun is detected and the SMC begins sending idles.

If the CM bit is set in the TxBD, the R bit is not cleared, so the CP can overwrite the buffer on its next access. For instance, if a single TxBD is initialized with the CM and W bits set, the buffer is sent continuously until R is cleared in the BD.

36.4.3 SMC Transparent Channel Reception Process

When the core enables the SMC receiver in transparent mode, it waits for synchronization before receiving data. Once synchronization is achieved, the receiver transfers the incoming data into memory according to the first RxBD in the table. Synchronization can be achieved in two ways. First, when the receiver is connected to a TDM channel, it can be synchronized to a time slot. Once the frame sync is received, the receiver waits for the first bit of its time slot to occur before reception begins. Data is received only during the time slots defined by the TSA. Secondly, when working with its own set of signals, the receiver starts reception when $\overline{\text{SMSYN}}x$ is asserted.

When the buffer full, the SMC clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the data buffer length, the SMC fetches the next BD; if it is empty, the

SMC continues transferring data to this BD's buffer. If the CM bit is set in the RxBD, the E bit is not cleared, so the CP can automatically overwrite the buffer on its next access.

36.4.4 Using $\overline{\text{SMSYN}}$ for Synchronization

The $\overline{\text{SMSYN}}$ signal offers a way to externally synchronize the SMC channel. This method differs somewhat from the synchronization options available in the SCCs and should be studied carefully. See Figure 36-11 for an example.

Once SMCMR[REN] is set, the first rising edge of SMCLK that finds $\overline{\text{SMSYN}}$ low causes the SMC receiver to achieve synchronization. Data starts being received or latched on the same rising edge of SMCLK that latched $\overline{\text{SMSYN}}$. This is the first bit of data received. The receiver does not lose synchronization again, regardless of the state of $\overline{\text{SMSYN}}$, until REN is cleared.

Once SMCMR[TEN] is set, the first rising edge of SMCLK that finds $\overline{\text{SMSYN}}$ low synchronizes the SMC transmitter which begins sending ones asynchronously from the falling edge of $\overline{\text{SMSYN}}$. After one character of ones is sent, if the transmit FIFO is loaded (the TxBD is ready with data), data starts being send on the next falling edge of SMCLK after one character of ones is sent. If the transmit FIFO is loaded later, data starts being sent after some multiple number of all-ones characters is sent.

Note that regardless of whether the transmitter or receiver uses $\overline{\text{SMSYN}}$, it must make glitch-free transitions from high-to-low or low-to-high. Glitches on $\overline{\text{SMSYN}}$ can cause errant behavior of the SMC.

The transmitter never loses synchronization again, regardless of the state of $\overline{\text{SMSYN}}$, until the TEN bit is cleared or an ENTER HUNT MODE command is issued.

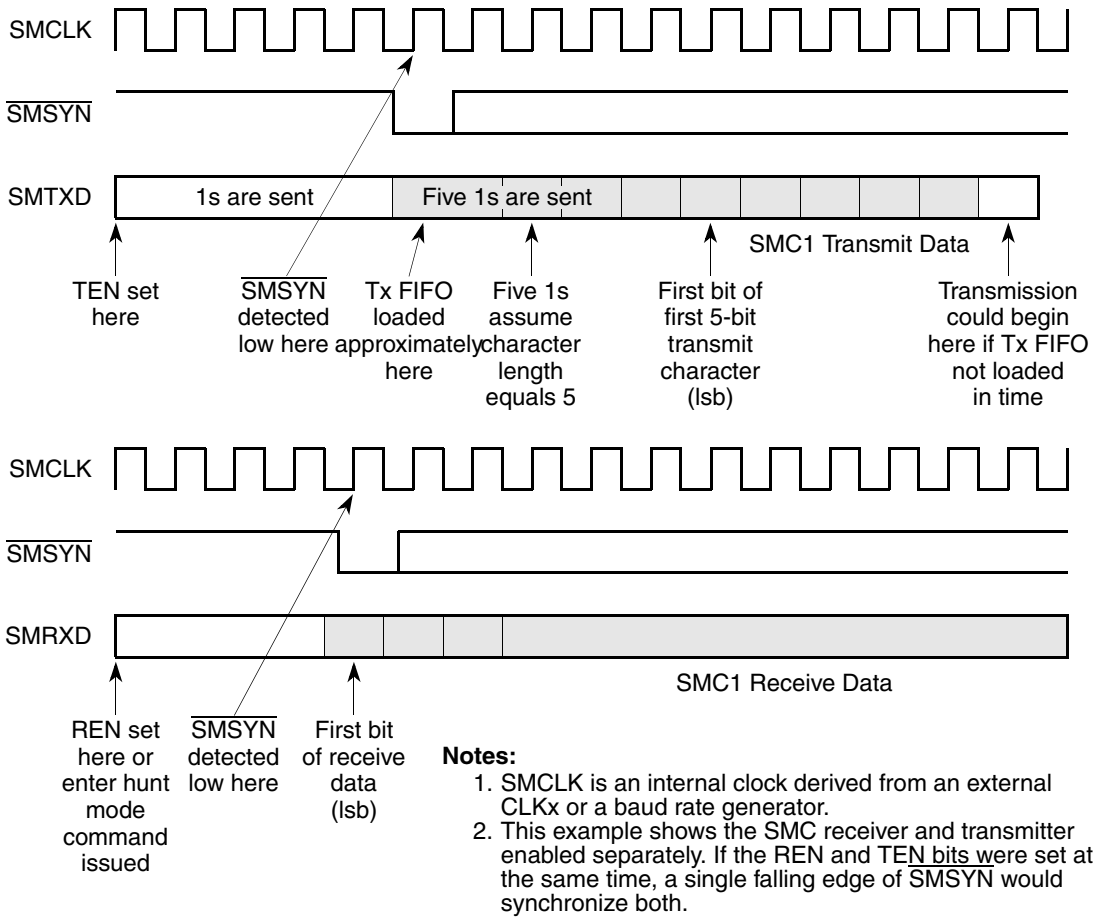


Figure 36-11. Synchronization with SMSYN*x*

If both SMCMR[REN] and SMCMR[TEN] are set, the first falling edge of SMSYN causes both the transmitter and receiver to achieve synchronization. The SMC transmitter can be disabled and re-enabled and SMSYN can be used again to resynchronize the transmitter itself. Section 36.2.4, "Disabling SMCs On-The-Fly," describes how to safely disable and re-enable the SMC. Simply clearing and setting TEN may be insufficient. The receiver can also be resynchronized this way.

36.4.5 Using the Time-Slot Assigner (TSA) for Synchronization

The TSA offers an alternative to using SMSYN to internally synchronize the SMC channel. This method is similar, except that the synchronization event is the first time-slot for this SMC receiver/transmitter after the frame sync indication rather than the falling edge of SMSYN. Chapter 23, "Serial Interface with Time-Slot Assigner," describes how to configure time slots. The TSA allows the SMC receiver and transmitter to be enabled simultaneously and synchronized separately; SMSYN does not provide this capability. Figure 36-12 shows synchronization using the TSA.

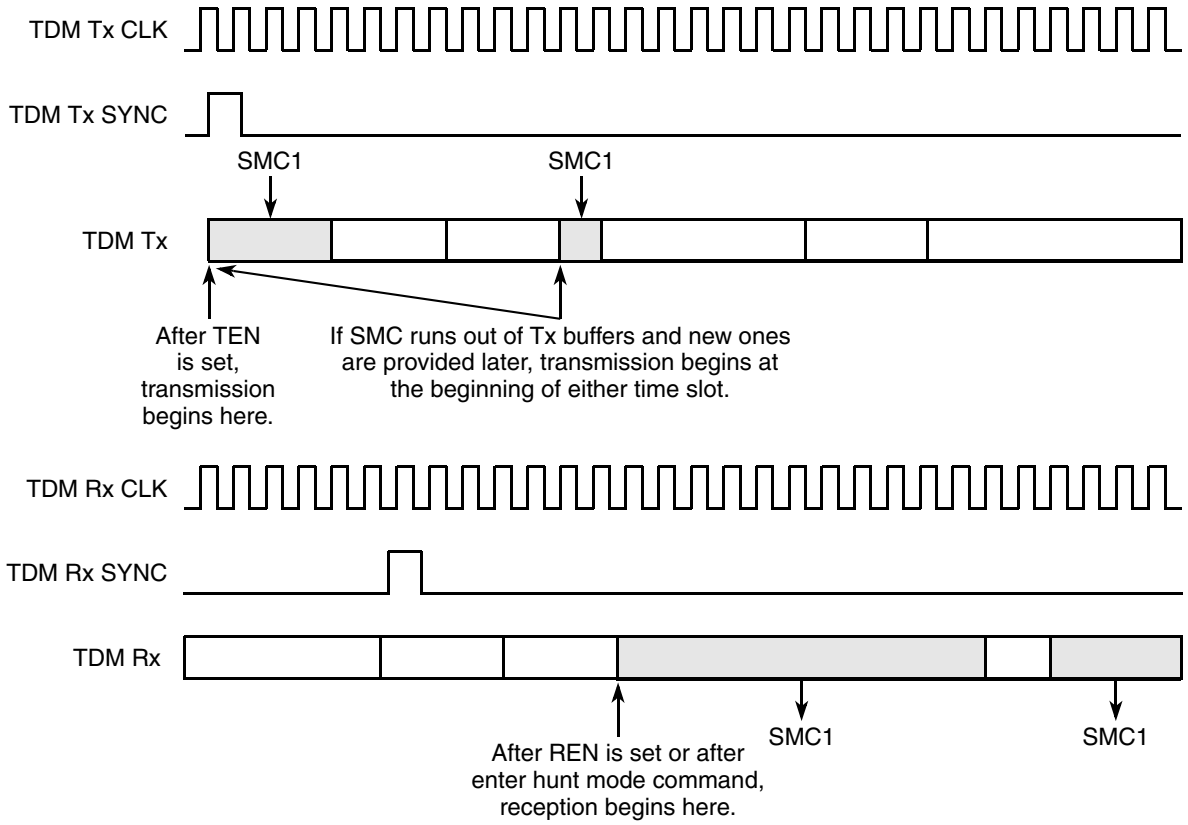


Figure 36-12. Synchronization with the TSA

Once SMCMR[REN] is set, the first time-slot after the frame sync causes the SMC receiver to achieve synchronization. Data is received immediately, but only during defined receive time slots. The receiver continues receiving data during its defined time slots until REN is cleared. If an ENTER HUNT MODE command is issued, the receiver loses synchronization, closes the buffer, and resynchronizes to the first time slot after the frame sync.

Once SMCMR[TEN] is set, the SMC waits for the transmit FIFO to be loaded before trying to achieve synchronization. When the transmit FIFO is loaded, synchronization and transmission begins depending on the following:

- If a buffer is made ready when the SMC2 is enabled, the first byte is placed in time slot 1 if CLSN is 8 and to slot 2 if CLSN is 16.
- If a buffer has its SMC enabled, then the first byte in the next buffer can appear in any time slot associated with this channel.
- If a buffer is ended with the L bit set, then the next buffer can appear in any time slot associated with this channel.

If the SMC runs out of transmit buffers and a new buffer is provided later, idles are sent in the gap between buffers. Data transmission from the later buffer begins at the start of an SMC time slot, but not necessarily the first time slot after the frame sync. So, to maintain a certain bit alignment beginning with the first time slot, make sure that at least one TxBD is always ready and that underruns do not occur. Otherwise, the SMC transmitter should be disabled and re-enabled. Section 36.2.4, “Disabling SMCs On-The-Fly,”

describes how to safely disable and re-enable the SMC. Simply clearing and setting TEN may not be enough.

36.4.6 SMC Transparent Commands

Table 36-10 describes transmit commands issued to the CPCR.

Table 36-10. SMC Transparent Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the SMCM, the channel is in transmit enable mode and polls the first BD. This command disables transmission of frames on the transmit channel. If the transparent controller receives this command while sending a frame, it stops after the contents of the FIFO are sent (up to 2 characters). The TBPTR is not advanced to the next BD, no new BD is accessed, and no new buffers are sent for this channel. The transmitter sends idles until a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Starts or resumes transmission from the current TBPTR in the channel TxBD table. When the channel receives this command, it polls the R bit in this BD. The SMC expects this command after a STOP TRANSMIT is issued. The channel in its mode register is disabled or after a transmitter error occurs.
INIT TX PARAMETERS	Initializes transmit parameters in this serial channel to reset state. Use only if the transmitter is disabled. The INIT TX AND RX PARAMETERS command resets transmit and receive parameters.

Table 36-11 describes receive commands issued to the CPCR.

Table 36-11. SMC Transparent Receive Commands

Command	Description
ENTER HUNT MODE	Forces the SMC to close the current receive BD if it is in use and to use the next BD for subsequent data. If the SMC is not receiving data, the buffer is not closed. Additionally, this command causes the receiver to wait for a resynchronization before reception resumes.
CLOSE RXBD	Forces the SMC to close the current receive BD if it in use and to use the next BD in the list for subsequent received data. If the SMC is not in the process of receiving data, no action is taken.
iNIT RX PARAMETERS	Initializes receive parameters in this serial channel to reset state. Use only if the receiver is disabled. The INIT TX AND RX PARAMETERS command resets receive and transmit parameters.

36.4.7 Handling Errors in the SMC Transparent Controller

The SMC uses BDs and the SMCE to report message send and receive errors.

Table 36-12. SMC Transparent Error Conditions

Error	Descriptions
Underrun	The channel stops sending the buffer, closes it, sets UN in the BD, and generates a TXE interrupt if it is enabled. The channel resumes sending after a RESTART TRANSMIT command. Underrun cannot occur between frames.
Overrun	The SMC maintains an internal FIFO for receiving data. If the buffer is in external memory, the CP begins programming the SDMA channel when the first character is received into the FIFO. If a FIFO overrun occurs, the SMC writes the received data character over the previously received character. The previous character and its status bits are lost. Then the channel closes the buffer, sets OV in the BD, and generates the RXB interrupt if it is enabled. Reception continues as normal.

36.4.8 SMC Transparent RxBD

Using BDs, the CP reports information about the received data for each buffer and closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

- An overrun error occurs.
- A full receive buffer is detected.
- The ENTER HUNT MODE command is issued.

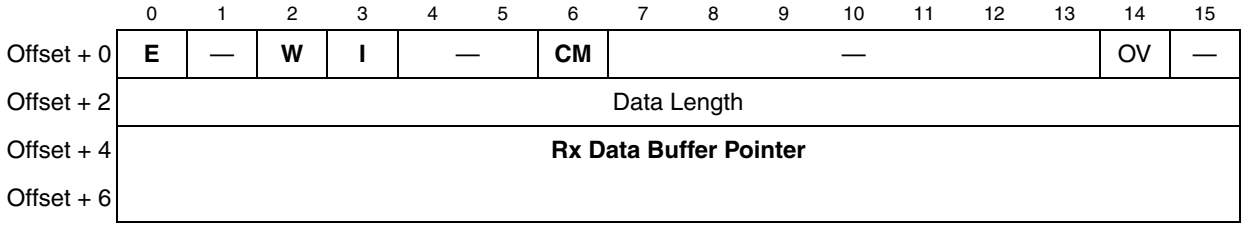


Figure 36-13. SMC Transparent RxBD

Table 36-13 describes SMC transparent RxBD fields.

Table 36-13. SMC Transparent RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The buffer is full or reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer is empty or is receiving data. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in RxBD table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to. The number of RxBDs is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is filled. 1 SMCE[RXB] is set when the CP completely fills this buffer indicating that the core must process the buffer. The RXB bit can cause an interrupt if it is enabled.
4–5	—	Reserved, should be cleared.
6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the buffer to be overwritten when the CP next accesses this BD. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7–13	—	Reserved, should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception. The CP writes OV after the received data is placed into the buffer.
15	—	Reserved, should be cleared.

Data length and buffer pointer fields are described in Section 28.3, “SCC Buffer Descriptors (BDs).”

36.4.9 SMC Transparent TxBD

Data is sent to the CP for transmission on an SMC channel by arranging it in buffers referenced by the channel TxBD table. The CP uses BDs to confirm transmission or indicate error conditions so the processor knows buffers have been serviced.

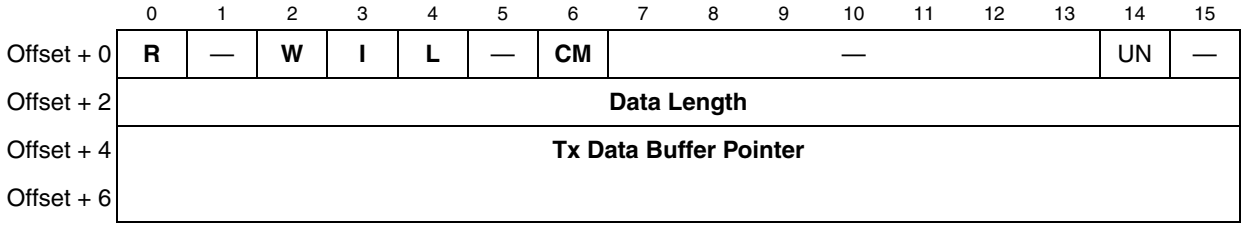


Figure 36-14. SMC Transparent TxBD

Table 36-14 describes SMC transparent TxBD fields.

Table 36-14. SMC Transparent TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready 0 The buffer is not ready for transmission. The BD and buffer can be updated. The CP clears R after the buffer is sent or after an error occurs. 1 The user-prepared data buffer is not sent or is being sent. BD fields cannot be updated if R is set.
1	—	Reserved, should be cleared.
2	**W**	Wrap (final BD in table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is programmable and determined by theW bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 SMCE[TXB] or SMCE[TXE] are set when the buffer is serviced. They can cause interrupts if they are enabled.
4	**L**	Last in message 0 The last byte in the buffer is not the last byte in the transmitted transparent frame. Data from the next transmit buffer (if ready) is sent immediately after the last byte of this buffer. 1 The last byte in this buffer is the last byte in the transmitted transparent frame. After this buffer is sent, the transmitter requires synchronization before the next buffer is sent.
5	—	Reserved, should be cleared.
6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear R after this BD is closed, allowing the buffer to be automatically resent when the CP accesses this BD again. However, the R bit is cleared if an error occurs during transmission, regardless of how CM is set.
7–13	—	Reserved, should be cleared.

Table 36-14. SMC Transparent TxBD Field Descriptions (continued)

Bits	Name	Description
14	UM	Underrun. Set when the SMC encounters a transmitter underrun condition while sending the buffer.
15	—	Reserved, should be cleared.

Data length represents the number of octets the CP should transmit from this buffer. It is never modified by the CP. The data length can be even or odd, but if the number of bits in the transparent character is greater than 8, the data length should be even. For example, to transmit three transparent 8-bit characters, the data length field should be initialized to 3. However, to transmit three transparent 9-bit characters, the data length field should be initialized to 6 because the three 9-bit characters occupy three half words in memory.

The data buffer pointer points to the first byte of the buffer. They can be even or odd, unless character length is greater than 8 bits, in which case the transmit buffer pointer must be even. For instance, the pointer to 8-bit transparent characters can be even or odd, but the pointer to 9-bit transparent characters must be even. The buffer can reside in internal or external memory.

36.4.10 SMC Transparent Event Register (SMCE)/Mask Register (SMCM)

The SMC event register (SMCE) generates interrupts and reports events recognized by the SMC channel. When an event is recognized, the SMC sets the corresponding SMCE bit. Interrupts are masked in the SMCM, which has the same format as the SMCE. SMCE bits are cleared by writing a 1 (writing 0 has no effect). Unmasked bits must be cleared before the CP clears the internal interrupt request. The SMCE and SMCM registers are displayed in Figure 36-15.

	0	1	2	3	4	5	6	7
Field	—			TXE	—	BSY	TXB	RXB
Reset	0							
R/W	R/W							
Offset								

Figure 36-15. SMC Transparent Event Register (SMCE)/Mask Register (SMCM)

Table 36-15 describes SMCE/SMCM fields.

Table 36-15. SMCE/SMCM Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3	TXE	Tx error. Set when an underrun error occurs on the transmitter channel.
4	—	Reserved, should be cleared.
5	BSY	Busy condition. Set when a character is received and discarded due to a lack of buffers. Reception begins after a new buffer is provided. Executing an ENTER HUNT MODE command makes the receiver wait for resynchronization.

Table 36-15. SMCE/SMCM Field Descriptions (continued)

Bits	Name	Description
6	TXB	Tx buffer. Set after a buffer is sent. If the L bit of the TxBD is set, TXB is set when the last character starts being sent. A one character-time delay is required to ensure that data is completely sent over the transmit signal. If the L bit of the TxBD is cleared, TXB is set when the last character is written to the transmit FIFO. A two character-time delay is required to ensure that data is completely sent.
7	RXB	Rx buffer. Set when a buffer is received (after the last character is written) on the SMC channel and its associated RxBD is now closed.

36.4.11 SMC Transparent NMSI Programming Example

The following example initializes the SMC1 transparent channel over its own set of signals. The CLK9 signal supplies the transmit and receive clocks; the $\overline{\text{SMSYN}x}$ signal is used for synchronization. (The SMC UART programming example uses a BRG configuration; see Section 36.3.12, "SMC UART Controller Programming Example.")

1. Configure the port D pins to enable SMTXD1, SMRXD1, and $\overline{\text{SMSYN1}}$. Set PPARD[7,8,9] and PDIRD[9]. Clear PDIRD[7,8] and PSORD[7,8,9].
2. Configure the port C pins to enable CLK9. Set PPARC[23]. Clear PDIRC[23] and PSORC[23].
3. Connect CLK9 to SMC1 using the CPM mux. Clear CMXSMR[SMC1] and program CMXSMR[SMC1CS] to 0b11.
4. In address 0x87FC, assign a pointer to the SMC1 parameter RAM.
5. Write RBASE and TBASE in the SMC parameter RAM to point to the RxBD and TxBD in the dual-port RAM. Assuming one RxBD at the beginning of the dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
6. Write 0x1D01_0000 to CPCR to execute the INIT RX AND TX PARAMETERS command.
7. Write RFCR and TFCR with 0x10 for normal operation.
8. Write MRBLR with the maximum bytes per receive buffer. Assuming 16 bytes MRBLR = 0x0010.
9. Initialize the RxBD assuming the buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
10. Initialize the TxBD assuming the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
11. Write 0xFF to SMCE1 to clear any previous events.
12. Write 0x13 to SMCM1 to enable all possible SMC1 interrupts.
13. Write 0x0000_1000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Write 0xFFFF_FFFF to the SIU interrupt pending register low (SIPNR_L) to clear events.
14. Write 0x3830 to the SMCMR to configure 8-bit characters, unreversed data, and normal operation (not loopback). The transmitter and receiver are not enabled yet.

15. Write 0x3833 to the SMCMR to enable the SMC transmitter and receiver. This additional write ensures that TEN and REN are enabled last.

After 5 bytes are sent, the TxBD is closed; after 16 bytes are received the receive buffer is closed. Any data received after 16 bytes causes a busy (out-of-buffers) condition since only one RxBD is prepared.

36.5 SMC in GCI Mode

The SMC can control the C/I and monitor channels of the GCI frame. When using the SCIT configuration of a GCI, one SMC can handle SCIT channel 0 and the other can handle SCIT channel 1. The main features of the SMC in GCI mode are as follows:

- Each SMC channel supports the C/I and monitor channels of the GCI (IOM-2) in ISDN applications
- Two SMCs support both sets of C/I and monitor channels in SCIT channels 0 and 1
- Full-duplex operation
- Local loopback and echo capability for testing

To use the SMC GCI channels properly, the TSA must be configured to route the monitor and C/I channels to the preferred SMC. Chapter 23, “Serial Interface with Time-Slot Assigner,” describes how to program this configuration. GCI mode is selected by setting SMCMR[SM] to 0b10. Section 36.2.1, “SMC Mode Registers (SMCMR1, SMCMR2),” describes other protocol-specific SMCMR bits.

36.5.1 SMC GCI Parameter RAM

The GCI parameter RAM differs from that for UART and transparent mode. The CP accesses each SMC’s GCI parameter table using a user-programmed pointer (SMC*x*_BASE) located in the parameter RAM; see Section 21.4.2, “Parameter RAM.” Each SMC GCI parameter RAM table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11, and 12). In GCI mode, parameter RAM contains the BDs instead of pointers to them. Compare Table 36-16 with Table 36-2 to see the differences. (In GCI mode, the SMC has no extra protocol-specific parameter RAM.)

Table 36-16. SMC GCI Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**M_RxBD**	Half word	Monitor channel RxBD. See Section 36.5.5, “SMC GCI Monitor Channel RxBD.”
0x02	**M_TxBD**	Half word	Monitor channel TxBD. See Section 36.5.6, “SMC GCI Monitor Channel TxBD.”
0x04	**CI_RxBD**	Half word	C/I channel RxBD. See Section 36.5.7, “SMC GCI C/I Channel RxBD.”
0x06	**CI_TxBD**	Half word	C/I channel TxBD. See Section 36.5.8, “SMC GCI C/I Channel TxBD.”
0x08	RSTATE[2]	Word	Rx/Tx internal state
0x0C	M_RxD[2]	Half word	Monitor Rx data
0x0E	M_TxD[2]	Half word	Monitor Tx data

Table 36-16. SMC GCI Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x10	CI_RxD[2]	Half word	C/I Rx data
0x12	CI_TxD[2]	Half word	C/I Tx data

1 From the pointer value programmed in SMC*x*_BASE: SMC1_BASE at 0x87FC, SMC2_BASE at 0x88FC.

2 RSTATE, M_RxD, M_TxD, CI_RxD, and CI_TxD do not need to be accessed by the user in normal operation, and are reserved for RISC use only.

36.5.2 Handling the GCI Monitor Channel

The following sections describe how the GCI monitor channel is handled.

36.5.2.1 SMC GCI Monitor Channel Transmission Process

Monitor channel 0 is used to exchange data with a layer 1 device (reading and writing internal registers and transferring of the S and Q bits). Monitor channel 1 is used for programming and controlling voice/data modules such as CODECs. The core writes the byte into the TxBD. The SMC sends the data on the monitor channel and handles the A and E control bits according to the GCI monitor channel protocol. The TIMEOUT command resolves deadlocks when errors in the A and E bit states occur on the data line.

36.5.2.2 SMC GCI Monitor Channel Reception Process

The SMC receives data and handles the A and E control bits according to the GCI monitor channel protocol. When the CP stores a received data byte in the SMC RxBD, a maskable interrupt is generated. A TRANSMIT ABORT REQUEST command causes the MPC8555E to send an abort request on the E bit.

36.5.3 Handling the GCI C/I Channel

The C/I channel is used to control the layer-1 device. The layer-2 device in the TE sends commands and receives indication to or from the upstream layer-1 device through C/I channel 0. In the SCIT configuration, C/I channel 1 is used to convey real-time status information between the layer-2 device and non–layer 1 peripheral devices (CODECs).

36.5.3.1 SMC GCI C/I Channel Transmission Process

The core writes the data byte into the C/I TxBD and the SMC transmits the data continuously on the C/I channel to the physical layer device.

36.5.3.2 SMC GCI C/I Channel Reception Process

The SMC receiver continuously monitors the C/I channel. When it recognizes a change in the data and this value is received in two successive frames, it is interpreted as valid data. This is called the double last-look method. The CP stores the received data byte in the C/I RxBD and a maskable interrupt is generated. If the SMC is configured to support SCIT channel 1, the double last-look method is not used.

36.5.4 SMC GCI Commands

The commands in Table 36-17 are issued to the CPCR.

Table 36-17. SMC GCI Commands

Command	Description
INIT TX AND RX PARAMETERS	Initializes transmit and receive parameters in the parameter RAM to their reset state. It is especially useful when switching protocols on a given serial channel.
TRANSMIT ABORT REQUEST	This receiver command can be issued when the MPC8555E implements the monitor channel protocol. When it is issued, the MPC8555E sends an abort request on the A bit.
TIMEOUT	This transmitter command can be issued when the MPC8555E implements the monitor channel protocol. It is usually issued because the device is not responding or A bit errors are detected. The MPC8555E sends an abort request on the E bit at the time this command is issued.

36.5.5 SMC GCI Monitor Channel RxBD

This BD, seen in Figure 36-16, is used by the CP to report information about the monitor channel receive byte.

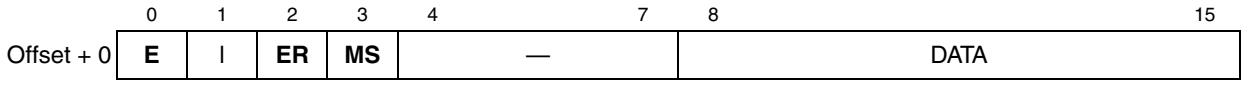


Figure 36-16. SMC Monitor Channel RxBD

Table 36-18 describes SMC monitor channel RxBD fields.

Table 36-18. SMC Monitor Channel RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The CP clears E when the byte associated with this BD is available to the core. 1 The core sets E when the byte associated with this BD has been read.
1	L	Last (EOM). Valid only for monitor channel protocol and is set when the EOM indication is received on the E bit. Note that when this bit is set, the data byte is invalid.
2	**ER**	Error condition. Valid only for monitor channel protocol. Set when an error occurs on the monitor channel protocol. A new byte is sent before the SMC acknowledges the previous byte.
3	**MS**	Data mismatch. Valid only for monitor channel protocol. Set when two different consecutive bytes are received; cleared when the last two consecutive bytes match. The SMC waits for the reception of two identical consecutive bytes before writing new data to the RxBD.
4–7	—	Reserved, should be cleared.
8–15	DATA	Data field. Contains the monitor channel data byte that the SMC received.

36.5.6 SMC GCI Monitor Channel TxBD

The CP uses this BD, shown in Figure 36-17, to report about the monitor channel transmit byte.

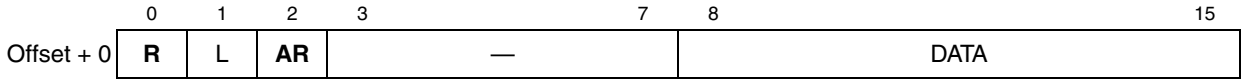


Figure 36-17. SMC Monitor Channel TxBD

Table 36-19 describes SMC monitor channel TxBD fields.

Table 36-19. SMC Monitor Channel TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready 0 Cleared by the CP after transmission. The TxBD is now available to the core. 1 Set by the core when the data byte associated with this BD is ready for transmission.
1	L	Last (EOM). Valid only for monitor channel protocol. When L = 1, the SMC first transmits the buffer data and then transmits the EOM indication on the E bit.
2	**AR**	Abort request. Valid only for monitor channel protocol. Set by the SMC when an abort request is received on the A bit. The transmitter sends the EOM on the E bit after receiving an abort request.
3–7	—	Reserved, should be cleared.
8–15	DATA	Data field. Contains the data to be sent by the SMC on the monitor channel.

36.5.7 SMC GCI C/I Channel RxBD

The CP uses this BD, seen in Figure 36-18, to report information about the C/I channel receive byte.

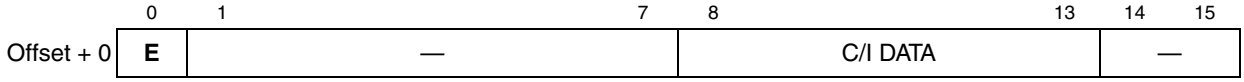


Figure 36-18. SMC C/I Channel RxBD

Table 36-20 describes SMC C/I channel RxBD fields.

Table 36-20. SMC C/I Channel RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 Cleared by the CP to indicate that the byte associated with this BD is available to the core. 1 The core sets E to indicate that the byte associated with this BD has been read. Note that additional data received is discarded until E bit is set.
1–7	—	Reserved, should be cleared.
8–13	C/I DATA	Command/indication data bits. For C/I channel 0, bits 10–13 contain the 4-bit data field and bits 8–9 are always written with zeros. For C/I channel 1, bits 8–13 contain the 6-bit data field.
14–15	—	Reserved, should be cleared.

36.5.8 SMC GCI C/I Channel TxBD

The CP uses this BD, as seen in Figure 36-19, to report about the C/I channel transmit byte.

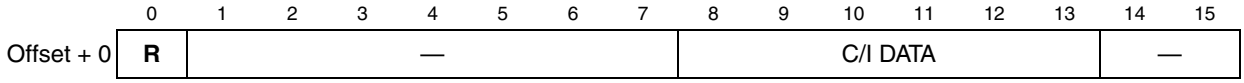


Figure 36-19. SMC C/I Channel TxBD

Table 36-21 describes SMC C/I channel TxBD fields.

Table 36-21. SMC C/I Channel TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 Cleared by the CP after transmission to indicate that the BD is available to the core 1 Set by the core when data associated with this BD is ready for transmission
1–7	—	Reserved, should be cleared.
8–13	C/I DATA	Command/indication data bits. For C/I channel 0, bits 10–13 hold the 4-bit data field (bits 8 and 9 are always written with zeros). For C/I channel 1, bits 8–13 contain the 6-bit data field.
14–15	—	Reserved, should be cleared.

36.5.9 SMC GCI Event Register (SMCE)/Mask Register (SMCM)

The SMCE generates interrupts and report events recognized by the SMC channel. When an event is recognized, the SMC sets its corresponding SMCE bit. SMCE bits are cleared by writing ones; writing zeros has no effect. SMCM has the same bit format as SMCE. Setting an SMCM bit enables, and clearing an SMCM bit disables, the corresponding interrupt. Unmasked bits must be cleared before the CP clears the internal interrupt request to the SIU interrupt controller. Figure 36-20 displays the SMCE/SMCM registers.

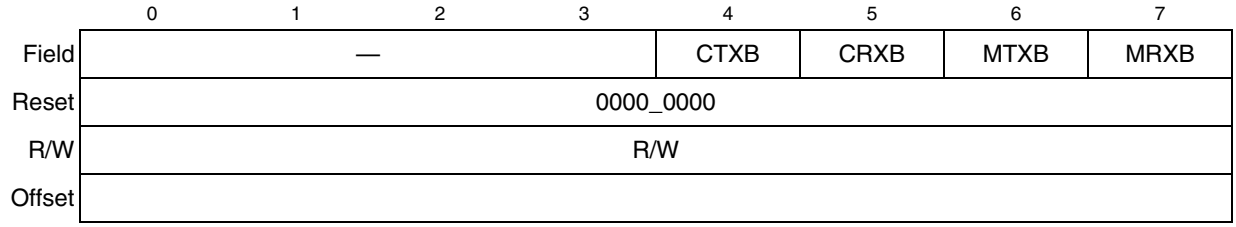


Figure 36-20. SMC GCI Event Register (SMCE)/Mask Register (SMCM)

Table 36-22 describes SMCE/SMCM fields.

Table 36-22. SMCE/SMCM Field Descriptions

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	CTXB	C/I channel buffer transmitted. Set when the C/I transmit buffer is now empty
5	CRXB	C/I channel buffer received. Set when the C/I receive buffer is full

Table 36-22. SMCE/SMCM Field Descriptions (continued)

Bits	Name	Description
6	MTXB	Monitor channel buffer transmitted. Set when the monitor transmit buffer is now empty
7	MRXB	Monitor channel buffer received. Set when the monitor receive buffer is full

Chapter 37
Fast Communications Controllers (FCCs)

The MPC8555E fast communications controllers (FCCs) are serial communications controllers (SCCs) optimized for synchronous high-rate protocols. FCC key features include the following:

- Supports HDLC/SDLC and totally transparent protocols
- FCC clocks can be derived from a baud-rate generator or an external signal.
- Supports $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem control signals
- Use of bursts to improve bus usage
- Multibuffer data structure for receive and transmit, external buffer descriptors (BDs) anywhere in system memory
- 192-byte FIFO buffers
- Full-duplex operation
- Fully transparent option for one half of an FCC (receiver/transmitter) while HDLC/SDLC protocol executes on the other half (transmitter/receiver)
- Echo and local loopback modes for testing
- Assuming a 100-MHz CPM clock, the FCCs support the following:
  - Full 10/100-Mbps Ethernet/IEEE 802.3x standard through an MII or RMII
  - Full 55-Mbps ATM segmentation and reassembly (SAR) through UTOPIA (on FCC1 and FCC2 only)
  - 45-Mbps (DS-3/E3 rates) HDLC and/or transparent data rates supported on each FCC

FCCs differ from SCCs as follows:

- No DPLL support
- No BISYNC, UART, or AppleTalk/LocalTalk support
- No HDLC bus

37.1 Overview

MPC8555E FCCs can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks. FCCs have many physical interface options such as interfacing to TDM buses, ISDN buses, standard modem interfaces, fast Ethernet interface (MII), and ATM interfaces (UTOPIA); see Chapter 23, “Serial Interface with Time-Slot Assigner,” and Chapter 41, “ATM Controller.” The FCCs are independent from the physical interface, but FCC logic formats and manipulates data from the physical interface. That is why the interfaces are described separately.

The FCC is described in terms of the protocol that it is chosen to run. When an FCC is programmed to a certain protocol, it implements a certain level of functionality associated with that protocol. For most protocols, this corresponds to portions of the link layer (layer 2 of the seven-layer OSI model). Many functions of the FCC are common to all of the protocols. These functions are described in the FCC description. Following that, the implementation details that differentiate protocols from one another are discussed, beginning with the transparent protocol. Thus, the reader should read from this point to the transparent protocol and then skip to the appropriate protocol. Since the FCCs use similar data structures across all protocols, the reader's learning time decreases dramatically after understanding the first protocol.

Each FCC supports a number of protocols—Ethernet, HDLC/SDLC, ATM, and totally transparent operation. Although the selected protocol usually applies to both the FCC transmitter and receiver, half of one FCC can run transparent operation while the other runs HDLC/SDLC protocol. The internal clocks (RCLK, TCLK) for each FCC can be programmed with either an external or internal source. The internal clocks originate from one of the baud-rate generators or one of the external clock signals. These clocks can be as fast as one-third the CPM clock frequency. See Chapter 23, “Serial Interface with Time-Slot Assigner.” However, the FCC’s ability to support a sustained bit stream depends on the protocol as well as on other factors.

NOTE

This clock ratio is based on the hardware architecture and does not ensure that an application will run at that speed. It is the responsibility of the system designer to check AC specifications of the I/O pins and determine the maximum frequency.

Each FCC can be connected to its own set of pins on the MPC8555E. This configuration, the nonmultiplexed serial interface, or NMSI, is described in Chapter 23, “Serial Interface with Time-Slot Assigner.” In this configuration, each FCC can support the standard modem interface signals ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$) through the appropriate port pins and the interrupt controller. Additional handshake signals can be supported with additional parallel I/O lines. The FCC block diagram is shown in Figure 37-1

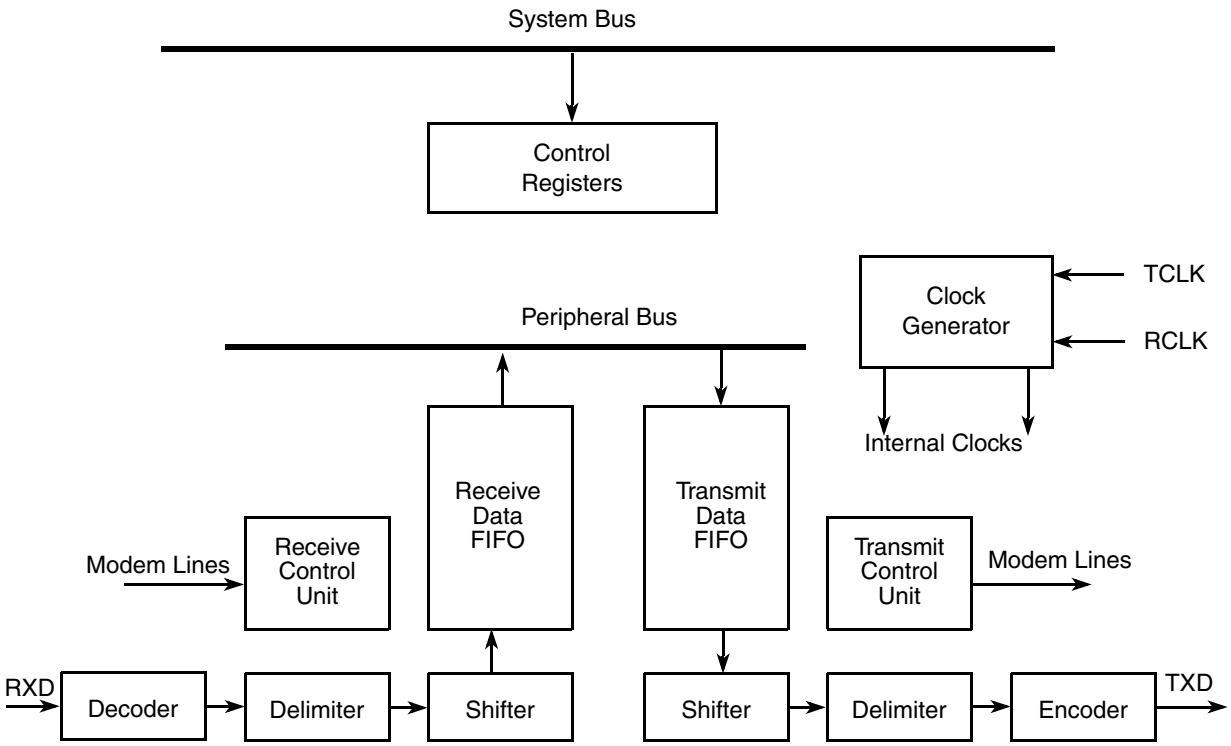


Figure 37-1. FCC Block Diagram

37.2 General FCC Mode Registers (GFMR*x*)

Each FCC contains a general FCC mode register (GFMR*x*) that defines common FCC options and selects the protocol to be run. The GFMR*x* are read/write registers cleared at reset. Figure 37-2 shows the GFMR format.

	0	1	2	3	4	5	6	7	8	9	15
Field	DIAG		TCI	TRX	TTX	CDP	CTSP	CDS	CTSS	—	
Reset	0000_0000_0000_0000										
R/W	R/W										
Offset	0x9_1300 (GFMR1), 0x9_1320 (GFMR2)										

	16	17	18	19	20	21	22	23	24	25	26	27	28	31
Field	SYNL		RTSM	RENC		REVD	TENC		TCRC		ENR	ENT	MODE	
Reset	0000_0000_0000_0000													
R/W	R/W													
Offset	0x9_1302 (GFMR1), 0x9_1322 (GFMR2)													

Figure 37-2. General FCC Mode Register (GFMR)

Table 37-1 describes GFMR fields.

Table 37-1. GFMR Register Field Descriptions

Bits	Name	Description
0–1	DIAG	Diagnostic mode. 00 Normal operation—Receive data enters through RXD, and transmit data is shifted out through TXD. The FCC uses the modem signals ($\overline{CD}$ and $\overline{CTS}$) to automatically enable and disable transmission and reception. Timings are shown in Section 37.12, "FCC Timing Control." 01 Local loopback mode—Transmitter output is connected internally to the receiver input, while the receiver and the transmitter operate normally. RXD is ignored. Data can be programmed to appear on TXD, or TXD can remain high by programming the appropriate parallel port register. $\overline{RTS}$ can be disabled in the appropriate parallel I/O register. The transmitter and receiver must use the same clock source, but separate CLK*x* pins can be used if connected to the same external clock source. If external loopback is preferred, program DIAG for normal operation and externally connect TXD and RXD. Then, physically connect the control signals ($\overline{RTS}$ connected to $\overline{CD}$, and $\overline{CTS}$ grounded) or set the parallel I/O registers so $\overline{CD}$ and $\overline{CTS}$ are permanently asserted to the FCC by configuring the associated $\overline{CTS}$ and $\overline{CD}$ pins as general-purpose I/O.; see Chapter 41, "ATM Controller." 10 Automatic echo mode—The channel automatically retransmits received data, using the receive clock provided. The receiver operates normally and receives data if $\overline{CD}$ is asserted. The transmitter simply transmits received data. In this mode, $\overline{CTS}$ is ignored. The echo function can also be accomplished in software by receiving buffers from an FCC, linking them to TxBDs, and transmitting them back out of that FCC. 11 Loopback and echo mode—Loopback and echo operation occur simultaneously. $\overline{CD}$ and $\overline{CTS}$ are ignored. Refer to the loopback bit description for clocking requirements. For TDM operation, the diagnostic mode is selected by SI*x*MR[SDM*x*]; see Section 23.6.2, "SI Mode Registers (SI2MR)."
2	TCI	Transmit clock invert 0 Normal operation 1 The FCC inverts the internal transmit clock. The edge on which the FCC outputs the data depends on the protocol: • In HDLC and Transparent mode, when TCI = 0, data is sent on the falling edge; when TCI = 1, on the rising edge. • In Ethernet mode, when TCI = 0, data is sent on the rising edge; when TCI = 1, on the falling edge.
3	TRX	Transparent receiver. The MPC8555E FCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is configured with the TTX and TRX bits instead of the MODE bits. This lets the user implement unique applications such as an FCC transmitter configured to HDLC and a receiver configured to totally transparent operation. To do this, program MODE = HDLC, TTX = 0, and TRX = 1. 0 Normal operation 1 The receiver operates in totally transparent mode, regardless of the protocol selected for the transmitter in the MODE bits. Note that full-duplex, totally transparent operation for an FCC is obtained by setting both TTX and TRX. Attempting to operate an FCC with Ethernet or ATM on its transmitter and transparent operation on its receiver causes erratic behavior. In other words, if the MODE = Ethernet or ATM, TTX must equal TRX.
4	TTX	Transparent transmitter. The MPC8555E FCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is configured with the TTX and TRX bits instead of the MODE bits. This lets the user implement unique applications, such as configuring an FCC receiver to HDLC and a transmitter to totally transparent operation. To do this, program MODE = HDLC, TTX = 1, and TRX = 0. 0 Normal operation 1 The transmitter operates in totally transparent mode, regardless of the receiver protocol selected in the MODE bits. Note that full-duplex totally transparent operation for an FCC is obtained by setting both TTX and TRX. Attempting to operate an FCC with Ethernet or ATM on its receiver and transparent operation on its transmitter causes erratic behavior. In other words, if GFMR[MODE] selects Ethernet or ATM, TTX must equal TRX.

Table 37-1. GFMR Register Field Descriptions (continued)

Bits	Name	Description
5	CDP	$\overline{CD}$ pulse (transparent mode only) 0 Normal operation (envelope mode). $\overline{CD}$ should envelope the frame; to negate $\overline{CD}$ while receiving causes a $\overline{CD}$ lost error. 1 Pulse mode. Once $\overline{CD}$ is asserted (high to low transition), synchronization has been achieved, and further transitions of $\overline{CD}$ do not affect reception. CDP must be set if this FCC is used with the TSA in transparent mode.
6	CTSP	$\overline{CTS}$ pulse 0 Normal operation (envelope mode). $\overline{CTS}$ should envelope the frame; to negate $\overline{CTS}$ while transmitting causes a $\overline{CTS}$ lost error. See Section 37.12, “FCC Timing Control.” 1 Pulse mode. $\overline{CTS}$ is asserted when synchronization is achieved; further transitions of $\overline{CTS}$ do not affect transmission. When running HDLC, the FCC samples $\overline{CTS}$ only once before sending the first frame after the transmitter is enabled (ENT = 1).
7	CDS	$\overline{CD}$ sampling 0 The $\overline{CD}$ input is assumed to be asynchronous with the data. The FCC synchronizes it internally before data is received. (This mode is not allowed in transparent mode when SYNL = 0b00.) 1 The $\overline{CD}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{CD}$ must transition while the receive clock is in the low state. When $\overline{CD}$ goes low, data is received. This is useful when connecting MPC8555Es in transparent mode, since it allows the $\overline{RTS}$ signal of one MPC8555E to be connected directly to the $\overline{CD}$ signal of another MPC8555E.
8	CTSS	$\overline{CTS}$ sampling 0 The $\overline{CTS}$ input is assumed to be asynchronous with the data. When it is internally synchronized by the FCC, data is sent after a delay of no more than two serial clocks. 1 The $\overline{CTS}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{CTS}$ must transition while the transmit clock is in the low state. As soon as $\overline{CTS}$ is low, data transmission begins. This mode is useful when connecting MPC8555E in transparent mode, because it allows the $\overline{RTS}$ signal of one MPC8555E to be connected directly to the $\overline{CTS}$ signal of another MPC8555E.
9--15	—	Reserved, should be 0.
16–17	SYNL	Sync length (transparent mode only). Determines the operation of an FCC receiver configured for totally transparent operation only. See Section 39.3.1, “In-Line Synchronization Pattern.” 00 The sync pattern in the FDSR is not used. An external sync signal is used instead ($\overline{CD}$ signal asserted: high to low transition). 01 Automatic sync (assumes always synchronized, ignores $\overline{CD}$ signal). 10 8-bit sync. The receiver synchronizes on an 8-bit sync pattern stored in the FDSR. Negation of $\overline{CD}$ causes CD lost error. 11 16-bit sync. The receiver synchronizes on a 16-bit sync pattern stored in the FDSR. Negation of $\overline{CD}$ causes CD lost error. Note that if SYNL = 1x, CDP should be cleared (not in $\overline{CD}$ pulse mode).
18	RTSM	$\overline{RTS}$ mode 0 Send idles between frames as defined by the protocol. $\overline{RTS}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{RTS}$ is asserted whenever the FCC is enabled.
19–20	RENC	Receiver decoding method. The user should set RENC = TENC in most applications. 00 NRZ 01 NRZI (one bit mode HDLC or transparent only) 1x Reserved

Table 37-1. GFMR Register Field Descriptions (continued)

Bits	Name	Description
21	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation 1 The totally transparent channels on this FCC (either the receiver, transmitter, or both, as defined by TTX and TRX) reverse bit order, transmitting the MSB of each octet first.
22–23	TENC	Transmitter encoding method. The user should set TENC = RENC in most applications. 00 NRZ 01 NRZI (one bit mode HDLC or transparent only) 1x Reserved
24-25	TCRC	Transparent CRC (totally transparent channel only). Selects the type of frame checking provided on the transparent channels of the FCC (either the receiver, transmitter, or both, as defined by TTX and TRX). This configuration selects a frame check type; the decision to send the frame check is made in the TxBD. Thus, it is not required to send a frame check in transparent mode. If a frame check is not used, the user can ignore any frame check errors generated on the receiver. 00 16-bit CCITT CRC (HDLC). (X16 + X12 + X5 + 1) 01 Reserved 10 32-bit CCITT CRC (Ethernet and HDLC) (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1) 11 Reserved
26	ENR	Enable receive. Enables the receiver hardware state machine for this FCC. 0 The receiver is disabled and any data in the receive FIFO buffer is lost. If ENR is cleared during reception, the receiver aborts the current character. 1 The receiver is enabled. ENR may be set or cleared regardless of whether serial clocks are present. Describes how to disable and re-enable an FCC. Note that the FCC provides other tools for controlling reception—the ENTER HUNT MODE command, CLOSE RXBD command, and RxBD[E].
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this FCC. 0 The transmitter is disabled. If ENT is cleared during transmission, the transmitter aborts the current character and TXD returns to idle state. Data in the transmit shift register is not sent. 1 The transmitter is enabled. ENT can be set or cleared, regardless of whether serial clocks are present. See Section 37.13, “Disabling the FCCs On-The-Fly,” for a description of the proper methods to disable and re-enable an FCC. Note that the FCC provides other tools for controlling transmission besides the ENT bit—the STOP TRANSMIT, GRACEFUL STOP TRANSMIT, and RESTART TRANSMIT commands, $\overline{\text{CTS}}$ flow control, and TxBD[R].
28–31	MODE	Channel protocol mode 0000 HDLC 0001 Reserved for RAM microcode 0010 Reserved 0011 Reserved for RAM microcode 0100 Reserved 0101 Reserved for RAM microcode 0110 Reserved 0111 Reserved for RAM microcode 1000 Reserved 1001 Reserved for RAM microcode 1010 ATM 1011 Reserved for RAM microcode 1100 Ethernet 11xx Reserved

37.3 General FCC Expansion Mode Register (GFEMR)

The general FCC expansion mode register (GFEMR) defines the expansion modes. It should be programmed according to the protocol used.

	0	1	2	3 … 7
Field	TIREM	LPB	CLK	—
Reset	0000_0000			
R/W	R/W			
Offset	0x9_1390 (GFEMR1), 0x9_13B0(GFEMR2)			

Figure 37-3. General FCC Expansion Mode Register (GFEMR)

Table 37-2 describes GFEMR*x* fields.

Table 37-2. GFEMR*x* Field Descriptions

Bit	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode: Internal rate for PHYs[0–3] is controlled only by FTIRR[0–3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode: PHYs[0–31] are controlled by FTIRR[0–3], FIRPER, FIRSR_HI, and FIRSR_LO. Underrun status for PHYs[0–31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode mixing of internal rate and external rate is not enabled.
1	LPB	RMII Loopback diagnostic mode (Ethernet mode): 0 Normal mode 1 Loopback mode
2	CLK	RMII reference clock rate for 50-MHz input clock from external oscillator (Ethernet mode): 0 50 MHz (for Fast Ethernet) 1 5 MHz (for 10BaseT)
3–7	—	Reserved, should be cleared.

37.4 FCC Protocol-Specific Mode Registers (FPSMR*x*)

The functionality of the FCC varies according to the protocol selected by GFMR[MODE]. Each FCC has an additional 32-bit, memory-mapped, read/write protocol-specific mode register (FPSMR) that configures them specifically for a chosen mode. The section for each specific protocol describes the FPSMR bits.

37.5 FCC Data Synchronization Registers (FDSR*x*)

Each FCC has a 16-bit, memory-mapped, read/write data synchronization register (FDSR), shown in Figure 37-2, that specifies the pattern used in the frame synchronization procedure of the synchronous protocols. In the totally transparent protocol, the FDSR should be programmed with the preferred SYNC pattern. For Ethernet protocol, it should be programmed with 0xD555. At reset, it defaults to 0x7E7E (two HDLC flags), so it does not need to be written for HDLC mode. The FDSR contents are always sent lsb first.

	0							7	8							15
Field	SYN2								SYN1							
Reset	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0
R/W	R/W															
Address	0x9_130C (FDSR1), 0x9_132C (FDSR2)															

Figure 37-4. FCC Data Synchronization Register (FDSR)

37.6 FCC Transmit-on-Demand Registers (FTODR*x*)

If no frame is being sent by the FCC, the CP periodically polls the R bit of the next TxBD to see if the user has requested a new frame/buffer to be sent. The polling algorithm depends on the FCC configuration, but occurs every 256 serial transmit clocks. The user, however, can request that the CP begin processing the new frame/buffer without waiting the normal polling time. For immediate processing, set the transmit-on-demand (TOD) bit in the transmit-on-demand register (FTODR), shown in Figure 37-5, after setting TxBD[R].

This feature, which decreases the transmission latency of the transmit buffer/frame, is particularly useful in LAN-type protocols where maximum interframe GAP times are limited by the protocol specification. Since the transmit-on-demand feature gives a high priority to the specified TxBD, it can conceivably affect the servicing of the other FCC FIFO buffers. Therefore, it is recommended that the transmit-on-demand feature be used only for a high-priority TxBD and when transmission on this FCC has not occurred for a given time period, which is protocol-dependent.

If a new TxBD is added to the BD table while preceding TxBDs have not completed transmission, the new TxBD is processed immediately after the older TxBDs are sent.

	0	1	15
Field	TOD	—	
Reset	0000_0000_0000_0000		
R/W	R/W		
Offset	0x9_1308 (FTODR1), 0x9_1328 (FTODR2)		

Figure 37-5. FCC Transmit-on-Demand Register (FTODR)

Fields in the FTODR are described in Table 37-3.

Table 37-3. FTODR Field Descriptions

Field	Name	Description
0	TOD	Transmit on demand 0 Normal polling 1 The CP gives high priority to the current TxBD and begins sending the frame does without waiting for the normal polling time to check TxBD[R]. TOD is cleared automatically.
1–15	—	Reserved, should be cleared.

37.7 FCC Buffer Descriptors

Data associated with each FCC is stored in buffers. Each buffer is referenced by a buffer descriptor (BD). All of the transmit BDs for an FCC are grouped into a TxBD circular table with a programmable length. Likewise, receive BDs form an RxBD table. The user can program the start address of the BD tables anywhere in system memory. See Figure 37-6.

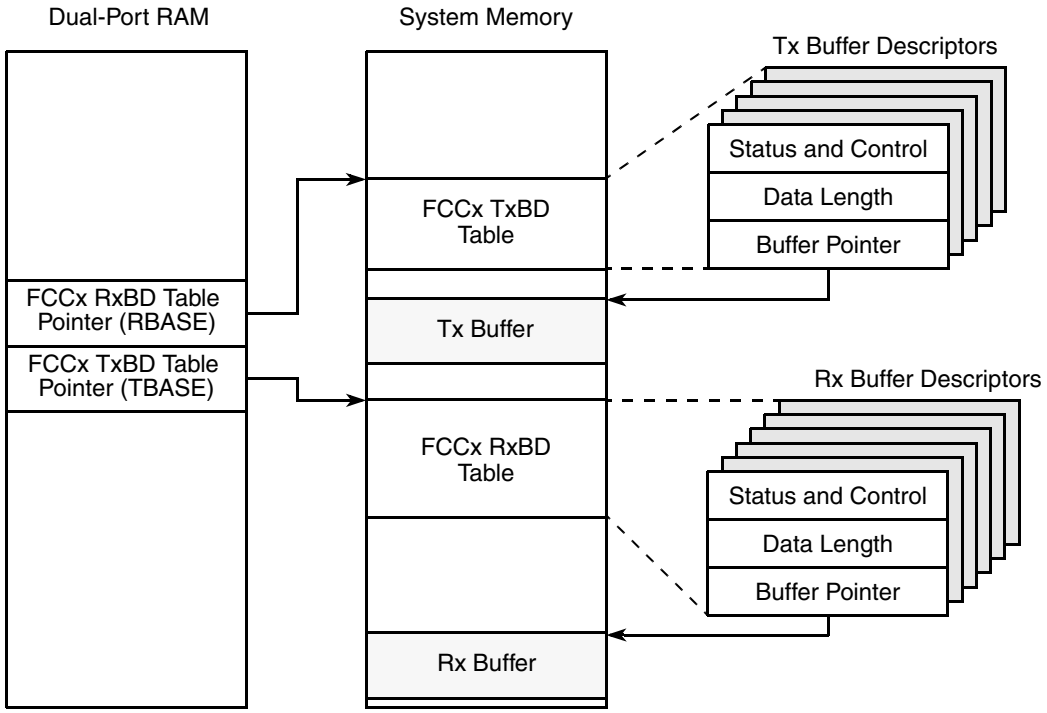


Figure 37-6. FCC Memory Structure

The format of transmit and receive BDs, shown in Figure 37-7, is the same for every FCC mode of operation except ATM mode; see Section 41.10.5, "ATM Controller Buffer Descriptors (BDs)." The first 16 bits in each BD contain status and control information, which differs for each protocol. The second 16 bits indicate the data buffer length in bytes (the wrap bit is the BD table length indicator). The remaining 32 bits contain the 32-bit address pointer to the actual buffer in memory.

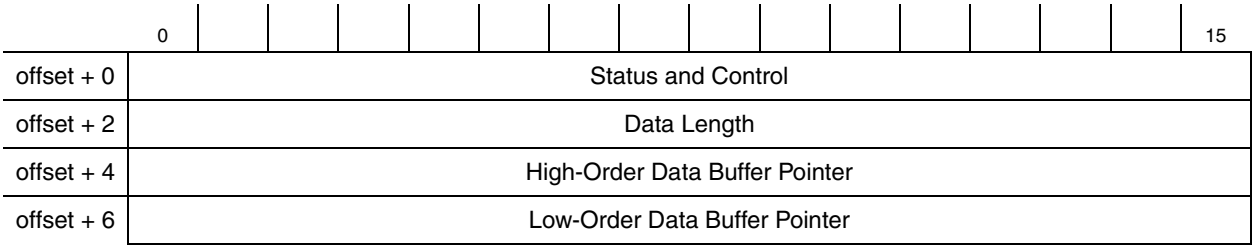


Figure 37-7. Buffer Descriptor Format

For frame-based protocols, a message can reside in as many buffers as necessary (transmit or receive). Each buffer has a maximum length of (64K–1) bytes. The CP does not assume that all buffers of a single frame are currently linked to the BD table. It does assume, however, that unlinked buffers are provided by the core soon enough to be sent or received. Failure to do so causes an error condition being reported by

the CP. An underrun error is reported in the case of transmit; a busy error is reported in the case of receive. Because BDs are prefetched, the receive BD table must always contain at least one empty BD to avoid a busy error; therefore, RxBD tables must always have at least two BDs.

The BDs and data buffers can be anywhere in the system memory.

The CP processes the TxBDs in a straightforward fashion. Once the transmit side of an FCC is enabled, it starts with the first BD in that FCC's TxBD table. When the CP detects that TxBD[R] is set, it begins processing the buffer. The CP detects that the BD is ready either by polling the R bit periodically or by the user writing to the FTODR. When the data from the BD has been placed in the transmit FIFO buffer, the CP moves on to the next BD, again waiting for the R bit to be set. Thus, the CP does no look-ahead BD processing, nor does it skip over BDs that are not ready. When the CP sees the wrap (W) bit set in a BD, it goes back to the beginning of the BD table after processing of the BD is complete.

After using a BD, the CP normally clears R (not-ready); thus, the CP does not use a BD again until the BD has been prepared by the core. Some protocols support continuous mode, which allows repeated transmission and for which the R bit remains set (always ready).

The CP uses RxBDs in a similar fashion. Once the receive side of an FCC is enabled, it starts with the first BD in the FCC's RxBD table. Once data arrives from the serial line into the FCC, the CP performs the required protocol processing on the data and moves the resultant data to the buffer pointed to by the first BD. Use of a BD is complete when no room is left in the buffer or when certain events occur, such as the detection of an error or end-of-frame. Regardless of the reason, the buffer is then said to be closed and additional data is stored using the next BD. Whenever the CP needs to begin using a BD because new data is arriving, it checks the E bit of that BD. This check is made on a prefetched copy of the current BD. If the current BD is not empty, it reports a busy error. However, it does not move from the current BD until it is empty. Because there is a periodic prefetch of the RxBD, the busy error may recur if the BD is not prepared soon enough.

When the CP sees the W bit set in a BD, it returns to the beginning of the BD table after processing of the BD is complete. After using a BD, the CP clears the E bit (not empty) and does not use a BD again until the BD has been processed by the core. However, in continuous mode, available to some protocols, the E bit remains set (always empty).

37.8 FCC Parameter RAM

Each FCC parameter RAM area begins at the same offset from each FCC base area. The protocol-specific portions of the FCC parameter RAM are discussed in the specific protocol descriptions. Table 37-4 shows portions common to all FCC protocols.

Some parameter RAM values must be initialized before the FCC is enabled; other values are initialized/written by the CP. Once initialized, most parameter RAM values do not need to be accessed by user software because most activity centers around the TxBDs and RxBDs rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled—after a STOP TRANSMIT command and before a RESTART TRANSMIT command or after the buffer/frame finishes

transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command.

- Rx parameter RAM can be written only when the receiver is disabled. Note the CLOSE RXBD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.
- See Section 37.13, "Disabling the FCCs On-The-Fly."

Some parameters in Table 37-4 are not described and are listed only to provide information for experienced users and for debugging. The user need not access these parameters in normal operation.

Table 37-4. FCC Parameter RAM Common to All Protocols Except ATM

Offset[1]	Name	Width	Description
0x00	RIPTR	Hword	Receive internal temporary data pointer. Used by microcode as a temporary buffer for data. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR. Must be 32-byte aligned and the size of the internal buffer must be 32 bytes, unless it is stated otherwise in the protocol specification.
0x02	TIPTR	Hword	Transmit internal temporary data pointer. Used by microcode as a temporary buffer for data. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR. Must be 32-byte aligned and the size of the internal buffer must be 32 bytes, unless it is stated otherwise in the protocol specification.
0x04	—	Hword	Reserved, should be cleared.
0x06	MRBLR	Hword	Maximum receive buffer length (a multiple of 32 for all modes). The number of bytes that the FCC receiver writes to a receive buffer before moving to the next buffer. The receiver can write fewer bytes to the buffer than MRBLR if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR value. Therefore, user-supplied buffers should be at least as large as the MRBLR. Note that FCC transmit buffers can have varying lengths by programming TxBD[Data Length], as needed, and are not affected by the value in MRBLR. MRBLR is not intended to be changed dynamically while an FCC is operating. Change MRBLR only when the FCC receiver is disabled.
0x08	RSTATE	Word	Receive internal state. The high byte, RSTATE[0–7], contains the function code register; see Section 37.8.1, "FCC Function Code Registers (FCRx)." RSTATE[8–31] is used by the CP and must be cleared initially.
0x0C	RBASE	Word	RxBD base address (must be divisible by eight). Defines the starting location in the memory map for the FCC RxBDs. This provides great flexibility in how FCC RxBDs are partitioned. By selecting RBASE entries for both FCCs and by setting the W bit in the last BD in each BD table, the user can select how many BDs to allocate for the receive side of every FCC. The user must initialize RBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled FCCs to overlap or erratic operation occurs.
0x10	RBDSTAT	Hword	RxBD status and control. Reserved for CP use only.
0x12	RBDLEN	Hword	RxBD data length. A down-count value initialized by the CP with MRBLR and decremented with every byte written by the SDMA channels.
0x14	RDPTR	Word	RxBD data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x18	TSTATE	Word	Tx internal state. The high byte, TSTATE[0–7], contains the function code register; see Section 37.8.1, "FCC Function Code Registers (FCRx)." TSTATE[8–31] is used by the CP and must be cleared initially.

Table 37-4. FCC Parameter RAM Common to All Protocols Except ATM (continued)

Offset[1]	Name	Width	Description
0x1C	TBASE	Word	TxBD base address (must be divisible by eight). Defines the starting location in the memory map for the FCC TxBDs. This provides great flexibility in how FCC TxBDs are partitioned. By selecting TBASE entries for both FCCs and by setting the W bit in the last BD in each BD table, the user can select how many BDs to allocate for the transmit side of every FCC. The user must initialize TBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled FCCs to overlap or erratic operation occurs.
0x20	TBDSTAT	Hword	TxBD status and control. Reserved for CP use only.
0x22	TBDLEN	Hword	TxBD data length. A down-count value initialized with the TxBD data length and decremented with every byte read by the SDMA channels.
0x24	TDPTR	Word	TxBD data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x28	RBPTR	Word	RxBD pointer. Points to the next BD that the receiver transfers data to when it is in idle state or to the current BD during frame processing. After a reset or when the end of the BD table is reached, the CP sets RBPTR = RBASE. Although the user need never write to RBPTR in most applications, the user can modify it when the receiver is disabled or when no receive buffer is in use.
0x2C	TBPTR	Word	TxBD pointer. Points either to the next BD that the transmitter transfers data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of the BD table is reached, the CP sets TBPTR = TBASE. Although the user need never write to TBPTR in most applications, the user can modify it when the transmitter is disabled or when no transmit buffer is in use (after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes transmission).
0x30	RCRC	Word	Temporary receive CRC
0x34	—	Word	Reserved
0x38	TCRC	Word	Temporary transmit CRC
0x3C	—	Word	First word of protocol-specific area

[1] Offset from FCC base: 0x8400 (FCC1), 0x8500 (FCC2); see Section 21.4.2, “Parameter RAM.”

37.8.1 FCC Function Code Registers (FCR*x*)

The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 37-8 shows the format of the transmit and receive function code registers, which reside at TSTATE[0–7] and RSTATE[0–7] in the FCC parameter RAM.

	0	1	2	3	4	5	6	7
Field	—	FCCP	GBL	BO		—	DTB	BDB

Figure 37-8. Function Code Register (FCR*x*)

FCR*x* fields are described in Table 37-5.

Table 37-5. FCR*x* Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1	FCCP	FCC priority 0 Disables CPM low request level to refer to FCCs 1 Enables CPM low request level to refer to FCCs
2	GBL	Global. Indicates whether the memory operation should be snooped 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering.Used to select the byte ordering of the buffer. If BO is modified on-the-fly, it takes effect at the start of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 0x Reserved 10 1x big endian
5	—	Reserved, should be cleared.
6	DTB	Indicates on what bus the data is located 0 On the system bus 1 On the local bus
7	BDB	Indicates on what bus the BDs are located 0 On the system bus 1 On the local bus

37.9 Interrupts From the FCCs

Interrupt handling for each of the FCC channels is configured on a global (per channel) basis in the interrupt pending register (SIPNR_L) and interrupt mask register (SIMR_L). One bit in each register is used to either mask, enable, or report an interrupt in an FCC channel. The interrupt priority between the FCCs is programmable in the CPM interrupt priority register (SCPRR_H). The interrupt vector register (SIVEC) indicates which pending channel has highest priority. Registers within the FCCs manage interrupt handling for FCC-specific events.

Events that can cause the FCC to interrupt the processor vary slightly among protocols and are described with each protocol. These events are handled independently for each channel by the FCC event and mask registers (FCCE and FCCM).

37.9.1 FCC Event Registers (FCCE*x*)

Each FCC has an FCC event register (FCCE) used to report events. On recognition of an event, the FCC sets its corresponding FCCE bit regardless of the corresponding mask bit. To the user it appears as a memory-mapped register that can be read at any time. Bits are cleared by writing ones; writing zeros has no effect on bit values. FCCE is cleared at reset. Fields of this register are protocol-dependent and are described in the respective protocol sections.

37.9.2 FCC Mask Registers (FCCM*x*)

Each FCC has a read/write FCC mask register (FCCM) used to enable or disable CP interrupts to the core for events reported in an event register (FCCE). Bit positions in FCCM are identical to those in FCCE. Note that an interrupt is generated only if the FCC interrupts are also enabled in the CPM interrupt controller; see Section 22.5.1.4, “CPM Interrupt Mask Registers (SIMR_H, SIMR_L).”

If an FCCM bit is zero, the CP does not proceed with its usual interrupt handling whenever that event occurs. Any time a bit in the FCCM register is set, a 1 in the corresponding bit in the FCCE register sets the FCC event bit in the interrupt pending register; see Section 22.5.1.4, “CPM Interrupt Mask Registers (SIMR_H, SIMR_L).”

37.9.3 FCC Status Registers (FCCS*x*)

Each FCC has an 8-bit, read/write FCC status register (FCCS) that lets the user monitor real-time status conditions (flags, idle) on the RXD line. It does not show the status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$; their real-time status is available in the appropriate parallel I/O port. (See Chapter 45, “Parallel I/O Ports.”)

37.10 FCC Initialization

The FCCs require a number of registers and parameters to be configured after a power-on reset. The following outline gives the proper sequence for initializing the FCCs, regardless of the protocol used.

1. Write the parallel I/O ports to configure and connect the I/O pins to the FCCs.
2. Write the appropriate port registers to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to be parallel I/O with interrupt capability or to connect directly to the FCC (if modem support is needed).
3. If the TSA is used, the SI must be configured. If the FCC is used in the NMSI mode, the CPM multiplexing logic (CMX) must still be initialized.
4. Write the GFMR, but do not write the ENT or ENR bits yet.
5. Write the FPSMR.
6. Write the FDSR.
7. Initialize the required values for this FCC in its parameter RAM.
8. Clear out any current events in FCCE, as needed.
9. Write the FCCM register to enable the interrupts in the FCCE register.
10. Write the SCPRR_H to configure the FCC interrupt priority.
11. Clear out any current interrupts in the SIPNR_L, if preferred.
12. Write the SIMR_L to enable interrupts to the CP interrupt controller.
13. Issue an INIT TX AND RX PARAMETERS command (with the correct protocol number).
14. Set GFMR[ENT] and GFMR[ENR].

The first RxBD’s empty bit must be set before the INIT RX COMMAND. However, TxBDs can have their ready bits set at any time. Notice that the CPCR does not need to be accessed after a power-on reset until an FCC is to be used. An FCC should be disabled and re-enabled after any dynamic change in its parallel I/O ports or serial channel physical interface configuration. A full reset using CPCR[RST] is a comprehensive reset that also can be used.

37.11 FCC Interrupt Handling

The following describes what usually occurs within an FCC interrupt handler:

1. When an interrupt occurs, read FCCE to determine interrupt sources. FCCE bits to be handled in this interrupt handler are normally cleared at this time.
2. Process the TxBDs to reuse them if the FCCE[TX,TXE] were set. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the FCC. Thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set.
3. Extract data from the RxBD if FCCE[RX, RXB, or RXF] is set. If the receive speed is fast or the interrupt delay is long, the FCC may have received more than one receive buffer. Thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the FCC prefetches BDs, the BD table must be big enough such that always there will be another empty BD to prefetch.
4. Clear FCCE.
5. Continue normal execution.

37.11.1 FCC Transmit Errors

There are four errors in the FCC transmitter that make it necessary for software to act on the transmitter before correct operation can continue. The errors are as follows:

- CTS-lost indication in HDLC transmitter (use re-initialization procedure)
- Underrun in any of the serial FCC transmitter protocols (use re-initialization procedure)
- Late collision in fast Ethernet transmitter (use recovery or re-initialization procedure)
- Expiration of retry limit in fast Ethernet (use recovery or re-initialization procedure)

In addition to status bits in the current TxBD, these errors are reported through the TXE event bit in the FCCE as a convenience for the user to implement error handling. TXE is a safe indication that a recovery or re-initialization procedure must be started.

37.11.1.1 Re-Initialization Procedure

1. Disable the FCC transmission by clearing GFMR[ENT].
2. Remember the TBPTR value taken from the FCC parameter RAM.
3. Issue an “INIT TX PARAMS” command using the CPCR.
4. Restore the remembered TBPTR into the FCC parameter RAM.
5. Adjust TxBD handling as described in Section Section 37.11.1.3, “Adjusting Transmitter BD Handling.”
6. Enable FCC transmission by setting GFMR[ENT].

37.11.1.2 Recovery Sequence

1. Adjust TxBD handling as described in Section Section 37.11.1.3, “Adjusting Transmitter BD Handling.”
2. Issue a “RESTART TX” command using the CPCR.

37.11.1.3 Adjusting Transmitter BD Handling

When a TXE event occurs, the TBPTR may already point beyond BDs still marked as ready due to internal pipelining. If the TBPTR is not adjusted, these BDs would be skipped while still being marked as ready. Software must determine if these BDs should be retransmitted or if they should be skipped, depending on the protocol and application needs. This requires the following steps:

1. From the current TBPTR value, search backwards over all (if any) BDs still marked as ready to find the first BD that has not been closed by the CPM. The search process should stop if the BD to be checked next is not ready or if it is the most recent BD marked as ready by the CPU transmit software. This is to avoid an endless loop in case the CPU software fills the BD ring completely.
2. A) For skipping BDs, manually close all BDs from the BD just found up to and including the BD just before TBPTR. Leave the TBPTR value untouched.

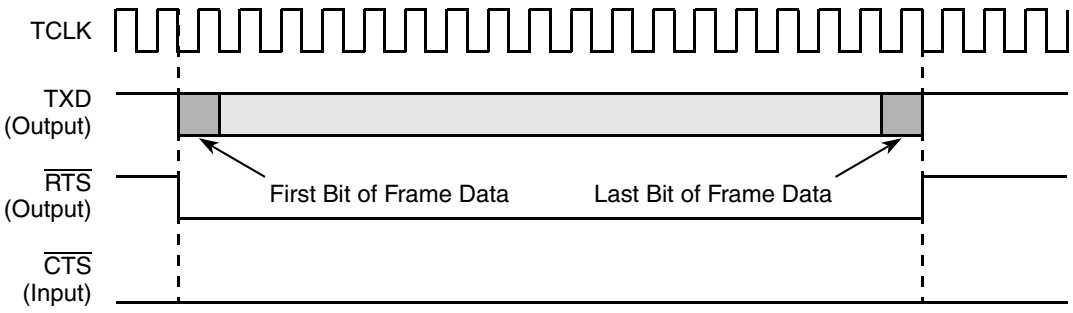
   B) For retransmitting BDs, change the TBPTR value to point to the BD just found.

37.12 FCC Timing Control

When GFMR[DIAG] is programmed to normal operation, $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are automatically controlled by the FCC. GFMR[TCI] is assumed to be cleared, which implies normal transmit clock operation.

$\overline{\text{RTS}}$ is asserted when FCC has data to transmit in the transmit FIFO and a falling transmit clock occurs. At this point, the FCC begins sending the data, once the appropriate conditions occur on $\overline{\text{CTS}}$. In all cases, the first bit of data is the start of the opening flag, or sync pattern.

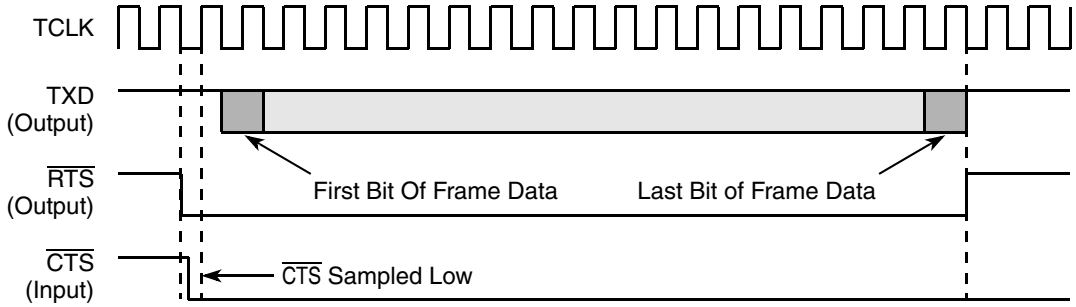
Figure 37-9 shows that the delay between $\overline{\text{RTS}}$ and data is 0 bit times, regardless of the setting of GFMR[CTSS]. This operation assumes that $\overline{\text{CTS}}$ is either already asserted to the FCC or is reprogrammed to be a parallel I/O line, in which case the $\overline{\text{CTS}}$ signal to the FCC is always asserted. $\overline{\text{RTS}}$ is negated one clock after the last bit in the frame.



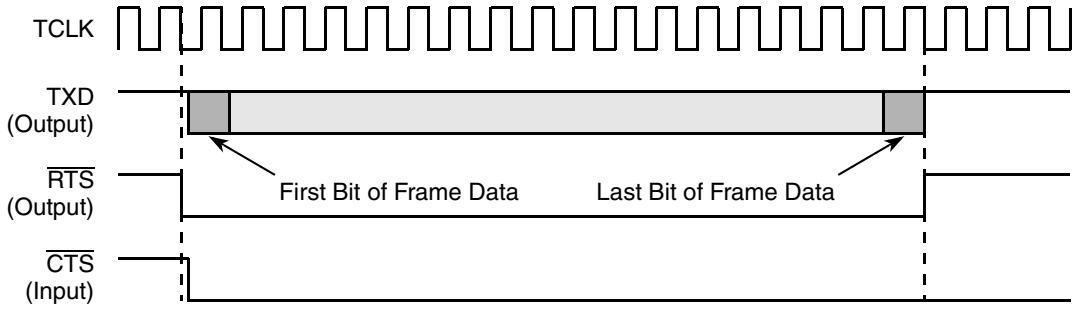
Note:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 37-9. Output Delay from $\overline{\text{RTS}}$ Asserted

If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ is asserted, the delays to the first bit of data depend on when $\overline{\text{CTS}}$ is asserted. Figure 37-10 shows that the delay between $\overline{\text{CTS}}$ and the data can be approximately 0.5 to 1 bit-times or no delay, depending on GFMR[CTSS].



Note:
1. GFMR[CTSS] = 0. CTSP is a don't care.



Note:
1. GFMR[CTSS] = 1. CTSP is a don't care.

Figure 37-10. Output Delay from $\overline{\text{CTS}}$ Asserted

If it is programmed to envelope the data, $\overline{\text{CTS}}$ must remain asserted during frame transmission or a $\overline{\text{CTS}}$ lost error occurs. The negation of $\overline{\text{CTS}}$ forces $\overline{\text{RTS}}$ high and the transmit data to the idle state. If GFMR[CTSS] = 0, the FCC must sample $\overline{\text{CTS}}$ before a $\overline{\text{CTS}}$ lost is recognized. Otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the $\overline{\text{CTS}}$ lost condition. See Figure 37-11.

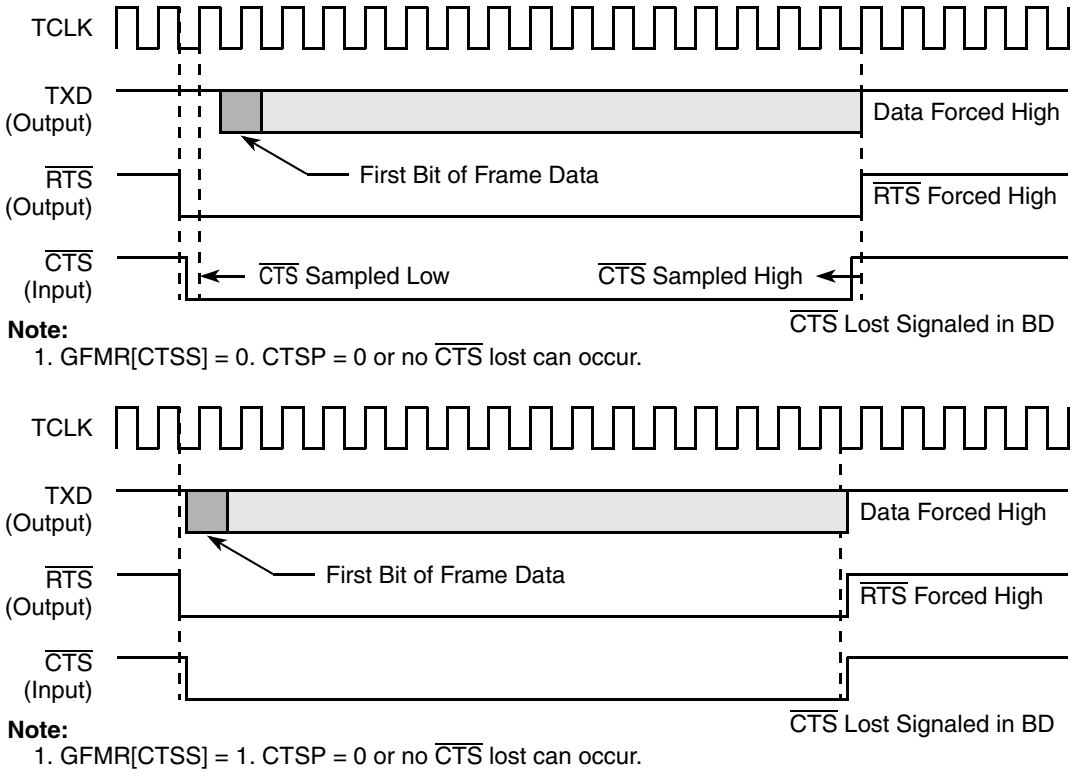
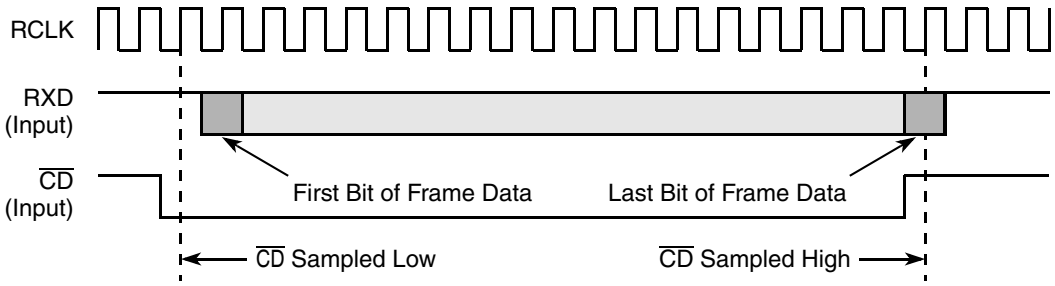


Figure 37-11. CTS Lost

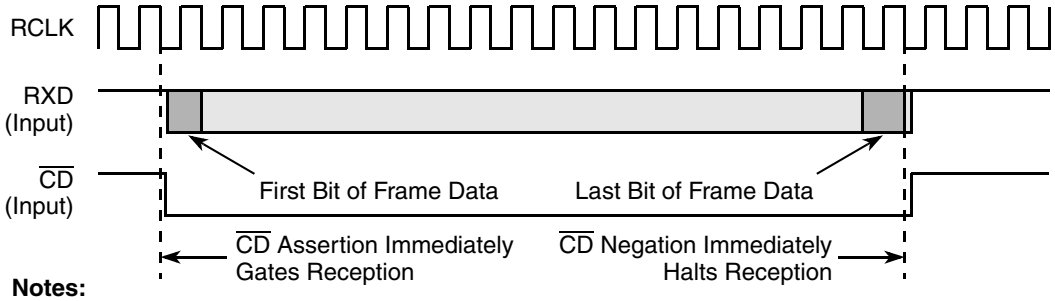
NOTE

Note that if GFMR[CTSS] = 1, all CTS transitions must occur while the transmit clock is low.

Reception delays are determined by CD as Figure 37-12 shows. If GFMR[CDS] = 0, CD is sampled on the rising receive clock edge before data is received. If GFMR[CDS] = 1, CD transitions immediately cause data to be gated into the receiver.



Notes:
1. GFMR[CDS] = 0. CDP = 0.
2. If $\overline{CD}$ is negated prior to the last bit of the receive frame, $\overline{CD}$ lost is signaled in the BD.
3. If CDP = 1, $\overline{CD}$ lost cannot occur and $\overline{CD}$ negation has no effect on reception.



Notes:
1. GFMR[CDS] = 1. CDP = 0.
2. If $\overline{CD}$ is negated prior to the last bit of the receive frame, $\overline{CD}$ lost is signaled in the BD.
3. If CDP = 1, $\overline{CD}$ lost cannot occur and $\overline{CD}$ negation has no effect on reception.

Figure 37-12. Using $\overline{CD}$ to Control Reception

If it is programmed to envelope data, $\overline{CD}$ must remain asserted during frame transmission or a $\overline{CD}$ lost error occurs. The negation of $\overline{CD}$ terminates reception. If [CDS] = 0, $\overline{CD}$ must be sampled by the FCC before a $\overline{CD}$ lost is recognized. Otherwise, the negation of $\overline{CD}$ immediately causes the $\overline{CD}$ lost condition.

Note that if GFMR[CDS] = 1, all $\overline{CD}$ transitions must occur while the receive clock is low.

37.13 Disabling the FCCs On-The-Fly

Unused FCCs can be temporarily disabled. In this case, a operation sequence is followed that ensures that any buffers in use are closed properly and that new data is transferred to or from a new buffer. Such a sequence is required if the parameters that must be changed are not allowed to be changed dynamically. If the register or bit description states that dynamic changes are allowed, the following sequences are not required and the register or bit may be changed immediately. In all other cases, the sequence should be used.

Modifying parameter RAM does not require the FCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CPCR[RST] to reset the entire CPM.

37.13.1 FCC Transmitter Full Sequence

For the FCC transmitter, the full disable and enable sequence is as follows.

1. Issue the STOP TRANSMIT command. This is recommended if the FCC is currently transmitting data because it stops transmission in an orderly way. If the FCC is not transmitting (no TxBDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and completed), then the STOP TRANSMIT command is not required. Furthermore, if the TBPTR is overwritten by the user or the INIT TX PARAMETERS command is executed, this command is not required.
2. Clear GFMR[ENT]. This disables the FCC transmitter and puts it in a reset state.
3. Make changes. The user can modify FCC transmit parameters, including the parameter RAM. To switch protocols or restore the FCC transmit parameters to their initial state, the INIT TX PARAMETERS command must be issued.
4. If an INIT TX PARAMETERS command was not issued in step 3, issue a RESTART TRANSMIT command.
5. Set GFMR[ENT]. Transmission begins using the TxBD that the TBPTR points to as soon as TxBD[R] = 1.

37.13.2 FCC Transmitter Shortcut Sequence

A shorter sequence is possible if the user prefers to reinitialize the transmit parameters to the state they had after reset. This sequence is as follows:

1. Clear GFMR[ENT].
2. Issue the INIT TX PARAMETERS command. Any additional changes can be made now.
3. Set GFMR[ENT].

37.13.3 FCC Receiver Full Sequence

The full disable and enable sequence for the receiver is as follows:

1. Clear GFMR[ENR]. Reception is aborted immediately, which disables the receiver of the FCC and puts it in a reset state.
2. Make changes. The user can modify the FCC receive parameters, including the parameter RAM. If the user prefers to switch protocols or restore the FCC receive parameters to their initial state, the INIT RX PARAMETERS command must be issued.
3. Issue the ENTER HUNT MODE command. This command is required if the INIT RX PARAMETERS command was not issued in step 2.
4. Set GFMR[ENR]. Reception begins immediately using the RxBD that the RBPTR points to if RxBD[E] = 1.

37.13.4 FCC Receiver Shortcut Sequence

A shorter sequence is possible if the user prefers to reinitialize the receive parameters to the state they had after reset. This sequence is as follows:

1. Clear GFMR[ENR].

2. Issue the INIT RX PARAMETERS command. Any additional changes can be made now.
3. Set GFMR[ENR].

37.13.5 Switching Protocols

A user can switch the protocol that the FCC is executing (HDLC) without resetting the board or affecting any other FCC by taking the following steps:

1. Clear GFMR[ENT] and GFMR[ENR].
2. Issue the INIT TX AND RX PARAMETERS command. This command initializes both transmit and receive parameters. Additional changes can be made in the GFMR to change the protocol.
3. Set GFMR[ENT] and GFMR[ENR]. The FCC is enabled with the new protocol.

37.14 Saving Power

Clearing an FCC's ENT and ENR bits minimizes its power consumption.

Chapter 38
FCC HDLC Controller

Layer 2 of the seven-layer OSI model is the data link layer (DLL), in which HDLC is one of the most common protocols. The framing structure of HDLC is shown in Figure 38-1. HDLC uses a zero insertion/deletion process (commonly known as bit stuffing) to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags. The HDLC frame is synchronous and therefore relies on the physical layer for a method of clocking and of synchronizing the transmitter/receiver.

Because the layer 2 frame can be transmitted over a point-to-point link, a broadcast network, or a packet-and-circuit switched system, an address field is needed for the frame's destination address. The length of this field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. For instance, SDLC and LAPB use an 8-bit address and SS#7 has no address field because it is used always in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within one device. It also defines a broadcast address. Some HDLC-type protocols also permit extended addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow-control number and defines the frame type (control or data). The exact use and structure of this field depends upon the protocol using the frame. Data is transmitted in the data field, which can vary in length depending upon the protocol using the frame. Layer 3 frames are carried in this data field.

Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16-bits long but can be as long as 32-bits. In HDLC, the lsb of each octet is transmitted first and the msb of the CRC is transmitted first.

When GFMR[MODE] selects HDLC mode, that FCC functions as an HDLC controller. When an FCC in HDLC mode is used with a nonmultiplexed modem interface, the FCC outputs are connected directly to the external pins. Modem signals can be supported through the appropriate port pins. The receive and transmit clocks can be supplied either externally or from the bank of baud-rate generators. The HDLC controller can also be connected to one of the TDM channels of the serial interface and used with the TSA. The HDLC controller consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with other FCCs. The user can allocate external buffer descriptors (BDs) for receive and transmit tasks so many frames can be sent or received without core intervention.

38.1 Key Features

Key features of the HDLC include the following:

- Flexible data buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (receive and transmit)
- Received frames threshold to reduce interrupt overhead

- Four address comparison registers with masks
- Maintenance of four 16-bit error counters
- Flag/abort/idle generation and detection
- Zero insertion/deletion
- 16- or 32-bit CRC-CCITT generation/checking
- Detection of nonoctet-aligned frames
- Detection of frames that are too long
- Programmable flags (0–15) between successive frames
- External BD table
- Up to T3 rate
- Support of time stamp mode for Rx frames
- Support of nibble mode HDLC (4 bits per clocks)

38.2 HDLC Channel Frame Transmission Processing

The HDLC transmitter is designed to work with almost no core intervention. When the core enables a transmitter, it starts sending flags or idles as programmed in the HDLC mode register (FPSMR). The HDLC controller polls the first BD in the transmit channel BD table. When there is a frame to transmit, the HDLC controller fetches the data (address, control, and information) from the first buffer and begins sending the frame after first inserting the user-specified minimum number of flags between frames. When the end of the current buffer is reached and TxBD[L] (last buffer in frame) is set, the FCC appends the CRC (if selected) and closing flag. In HDLC, the lsb of each octet and the msb of the CRC are sent first. Figure 38-1 shows a typical HDLC frame.

Opening Flag	Address	Control	Information (Optional)	CRC	Closing Flag
8 bits	16 bits	8 bits	8*n* bits	16 bits	8 bits

Figure 38-1. HDLC Framing Structure

After the closing flag is sent, the HDLC controller writes the frame status bits into the BD and clears the R bit. When the end of the current BD is reached and the L (last) bit is not set (working in multibuffer mode), only the R bit is cleared. In either mode, an interrupt can be issued if the I bit in the TxBD is set. The HDLC controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each buffer, after a specific buffer, after each frame, or after a number of frames.

To rearrange the transmit queue before the CP has sent all buffers, issue the STOP TRANSMIT command. This can be useful for sending expedited data before previously linked buffers or for error situations. When receiving the STOP TRANSMIT command, the HDLC controller aborts the current frame transmission and starts transmitting idles or flags. When the HDLC controller is given the RESTART TRANSMIT command, it resumes transmission. To insert a high-priority frame without aborting the current frame, the GRACEFUL STOP TRANSMIT command can be issued. A special interrupt (GRA) can be generated in the event register when the current frame is complete.

38.3 HDLC Channel Frame Reception Processing

The HDLC receiver is designed to work with almost no core intervention and can perform address recognition, CRC checking, and maximum frame length checking. The received frame is available for any HDLC-based protocol. When the core enables a receiver, the receiver waits for an opening flag character. When it detects the first byte of the frame, the HDLC controller compares the frame address against the user-programmable addresses. The user has four 16-bit address registers and an address mask available for address matching. The HDLC controller compares the received address field to the user-defined values after masking with the address mask. The HDLC controller can also detect broadcast (all ones) address frames if one address register is written with all ones.

If a match is detected, the HDLC controller checks the prefetched BD; if it is empty, it starts transferring the incoming frame to the BD's associated buffer. When the buffer is full, the HDLC controller clears BD[E] and generates an interrupt if BD[I] = 1. If the incoming frame is larger than the buffer, the HDLC controller fetches the next BD in the table and, if it is empty, continues transferring the frame to the associated buffer.

During this process, the HDLC controller checks for frames that are too long. When the frame ends, the CRC field is checked against the recalculated value and written to the buffer. The data length written to the last BD in the HDLC frame is the length of the entire frame. This enables HDLC protocols that lose frames to correctly recognize a frame-too-long condition.

The HDLC controller then sets the last buffer in frame bit, writes the frame status bits into the BD, and clears the E bit and fetches the next BD. The HDLC controller then generates a maskable interrupt, indicating that a frame was received and is in memory. The HDLC controller then waits for a new frame. Back-to-back frames can be received separated only by a single shared flag.

The user can configure the HDLC controller not to interrupt the core until a specified number of frames have been received. This is configured in the received frames threshold (RFTHR) location of the parameter RAM. This function can be combined with a timer to implement a time-out if fewer than the threshold number of frames are received.

38.4 HDLC Parameter RAM

When an FCC operates in HDLC mode, the protocol-specific area of the FCC parameter RAM is mapped with the HDLC-specific parameters in Table 38-1.

Table 38-1. FCC HDLC-Specific Parameter RAM Memory Map

Offset[1]	Name[2]	Width	Description
0x3C	—	2 Words	Reserved
0x44	**C_MASK**	Word	CRC constant. For the 16-bit CRC-CCITT, initialize C_MASK to 0x0000_F0B8. For the 32-bit CRC-CCITT, initialize C_MASK to 0xDEBB_20E3.
0x48	**C_PRES**	Word	CRC preset. For the 16-bit CRC-CCITT, initialize C_PRES to 0x0000_FFFF. For the 32-bit CRC-CCITT, initialize C_PRES to 0xFFFF_FFFF.
0x4C	**DISFC**[3]	Hword	Discard frame counter. Counts error-free frames discarded due to lack of buffers.

Table 38-1. FCC HDLC-Specific Parameter RAM Memory Map (continued)

Offset[1]	Name[2]	Width	Description
0x4E	**CRCEC** [3]	Hword	CRC error counter. Counts frames not addressed to the user or frames received in the BSY condition, but does not include overrun, $\overline{CD}$ lost, or abort errors.
0x50	**ABTSC** [3]	Hword	Abort sequence counter
0x52	**NMARC** [3]	Hword	Nonmatching address Rx counter. Counts nonmatching addresses received (error-free frames only). See the HMASK and HADDR[1–4] parameter description.
0x54	MAX_CNT	Word	Max_length counter. Temporary decrementing counter that tracks frame length.
0x58	**MFLR**	Hword	Max frame length register. If the HDLC controller detects an incoming HDLC frame that exceeds the user-defined value in MFLR, the rest of the frame is discarded and the LG (Rx frame too long) bit is set in the last BD belonging to that frame. The HDLC controller waits for the end of the frame and then reports the frame status and length in the last RxBD. MFLR includes all in-frame bytes between the opening and closing flags (address, control, data, and CRC).
0x5A	**RFTHR**	Hword	Received frames threshold. Used to reduce the interrupt overhead that might otherwise occur when a series of short HDLC frames arrives, each causing an RXF interrupt. By programming RFTHR, the user lowers the frequency of RXF interrupts, which occur only when the RFTHR value is reached. Note that the user should provide enough empty RxBDs to receive the number of frames specified in RFTHR.
0x5C	**RFCNT**	Hword	Received frames count. A decrementing counter used to implement this feature. Initialize this counter with RFTHR.
0x5E	**HMASK**	Hword	HMASK and HADDR[1–4]. The HDLC controller reads the frame address from the HDLC receiver, checks it against the four address register values, and masks the result with HMASK. In HMASK, a 1 represents a bit position for which address comparison should occur; 0 represents a masked bit position. When addresses match, the address and subsequent data are written into the buffers. When addresses do not match and the frame is error-free, the nonmatching address received counter (NMARC) is incremented. Note that for 8-bit addresses, mask out (clear) the eight high-order bits in HMASK. The eight low-order bits and HADDRx should contain the address byte that immediately follows the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDRx should contain 0xAA68 and HMASK should contain 0xFFFF. See Figure 38-2.
0x60	**HADDR1**	Hword	
0x62	**HADDR2**	Hword	
0x64	**HADDR3**	Hword	
0x66	**HADDR4**	Hword	
0x68	TS_TMP	Hword	Temporary storage
0x6A	TMP_MB	Hword	Temporary storage

1 Offset from FCC base: 0x8400 (FCC1) and 0x8500 (FCC2); see Section 21.4.2, “Parameter RAM.”

2 Boldfaced entries must be initialized by the user.

3 DISFC, CRCEC, ABTSC, and NMARC—These 16-bit (modulo 216) counters are maintained by the CP. The user can initialize them while the channel is disabled.

Figure 38-2 shows an example of using HMASK and HADDR[1–4].

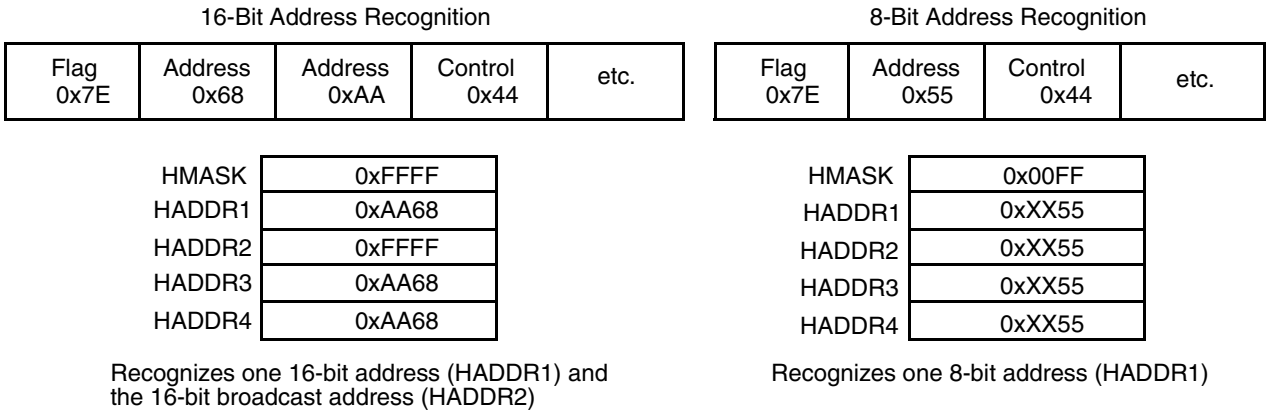


Figure 38-2. HDLC Address Recognition Example

38.5 Programming Model

The core configures each FCC to operate in the protocol specified in GFMR[MODE]. The HDLC controller uses the same data structure as other modes. This data structure supports multibuffer operation and address comparisons.

38.5.1 HDLC Command Set

The transmit and receive commands are issued to the CPCR; see Section 21.3, "Command Set."

Table 38-2 describes the transmit commands that apply to the HDLC controller.

Table 38-2. Transmit Commands

Command	Description
STOP TRANSMIT	After the hardware or software is reset and the channel is enabled in the FCC mode register, the channel is in transmit enable mode and starts polling the first BD in the table every 256 transmit clocks (immediately if FTODR[TOD] = 1). STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the HDLC controller during frame transmission, transmission is aborted after a maximum of 64 additional bits are sent and the transmit FIFO buffer is flushed. The TBPTR is not advanced, no new BD is accessed, and no new frames are sent for this channel. The transmitter sends an abort sequence consisting of 0x7F (if the command was given during frame transmission) and begins sending flags or idles, as indicated by the HDLC mode register. Note that if FPSMR[MFF] = 1, one or more small frames can be flushed from the transmit FIFO buffer. The GRACEFUL STOP TRANSMIT command can be used to avoid this.
GRACEFUL STOP TRANSMIT	Used to stop transmission smoothly rather than abruptly, as performed by the regular STOP TRANSMIT command. It stops transmission after the current frame finishes sending or immediately if no frame is being sent. FCCE[GRA] is set once transmission has stopped. Then the HDLC transmit parameters (including BDs) can be modified. The TBPTR points to the next TxBD in the table. Transmission begins once the R bit of the next BD is set and the RESTART TRANSMIT command is issued.

Table 38-2. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Enables character transmission on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command is issued and the channel in its FCC mode register is disabled, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error (underrun or $\overline{\text{CTS}}$ lost with no automatic frame retransmission). The HDLC controller resumes sending from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in this serial channel parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Notice that the INIT TX AND RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Table 38-3 describes the receive commands that apply to the HDLC controller.

Table 38-3. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel is enabled in the FCC mode register, the channel is in receive enable mode and uses the first BD in the table. The ENTER HUNT MODE command is generally used to force the HDLC receiver to abort reception of the current frame and enter the hunt mode. In hunt mode, the HDLC controller continually scans the input data stream for the flag sequence. After receiving the command, the current receive buffer is closed, the error status flags and length field are cleared, RxBD[E] (the empty bit) is set, and the CRC calculation is reset. Further frame reception uses the current RxBD.
INIT RX PARAMETERS	Initializes all the receive parameters in this serial channel parameter RAM to their reset state and should be issued only when the receiver is disabled. Notice that the INIT TX AND RX PARAMETERS command resets both receive and transmit parameters.

38.5.2 HDLC Error Handling

The HDLC controller reports frame reception and transmission error conditions using the channel BDs, error counters, and HDLC event register (FCCE). Table 38-4 describes HDLC transmission errors, which are reported through the TxBD.

Table 38-4. HDLC Transmission Errors

Error	Description
Transmitter Underrun	When this error occurs, the channel terminates buffer transmission, transmit an ABORT sequence (a sequence which will generate CRC error on the frame), closes the buffer, sets the underrun (U) bit in the BD, and generates the TXE interrupt if it is enabled. The channel resumes transmission after receiving the RESTART TRANSMIT command.
$\overline{\text{CTS}}$ Lost during Frame Transmission	When this error occurs, the channel terminates buffer transmission, closes the buffer, sets TxBD[CT], and generates a TXE interrupt (if it is enabled). The channel resumes transmission after receiving the RESTART TRANSMIT command.

Table 38-5 describes HDLC reception errors, which are reported through the RxBD.

Table 38-5. HDLC Reception Errors

Error	Description
Overrun Error	The HDLC controller maintains an internal FIFO buffer for receiving data. The CP begins programming the SDMA channel and updating the CRC whenever data is received in the FIFO buffer. When a receive FIFO overrun occurs, the channel writes the received data byte to the internal FIFO buffer over the previously received byte. The previous byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates the RXF interrupt if it is enabled. The receiver then enters hunt mode. Even if the overrun occurs during a frame whose address is not matched in the address recognition logic, an RxBD with data length two is opened to report the overrun and the RXF interrupt is generated if it is enabled.
$\overline{\text{CD}}$ Lost During Frame Reception	When this error occurs, the channel terminates frame reception, closes the buffer, sets RxBD[CD], and generates the RXF interrupt if it is enabled. This error has highest priority. The rest of the frame is lost and other errors are not checked in that frame. At this point, the receiver enters hunt mode. If $\overline{\text{CD}}$ is Lost during the first 8 serial bits it will not be reported as $\overline{\text{CD}}$ Lost error and there will be no indication of error.
Abort Sequence	The HDLC controller detects an abort sequence when seven or more consecutive ones are received. When this error occurs and the HDLC controller receives a frame, the channel closes the buffer by setting RxBD[AB] and generates the RXF interrupt (if enabled). The channel also increments the abort sequence counter. The CRC and nonoctet error status conditions are not checked on aborted frames. The receiver then enters hunt mode. When an abort sequence is received, the user is given no indication that an HDLC controller is not currently receiving a frame.
Nonoctet Aligned Frame	When this error occurs, the channel writes the received data to the data buffer, closes the buffer, sets the Rx nonoctet aligned frame bit RxBD[NO], and generates the RXF interrupt (if it is enabled). The CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. An immediate back-to-back frame is still received. The nonoctet data portion may be derived from the last byte in the buffer by finding the least-significant set bit, which marks the end of valid data as follows: msb | | | | | | | lsb Valid data | 1 | 0 | 0 | 0
CRC Error	When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets RxBD[CR], and generates the RXF interrupt (if it is enabled). The channel also increments the CRC error counter. After receiving a frame with a CRC error, the receiver enters hunt mode. An immediate back-to-back frame is still received. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

38.6 HDLC Mode Register (FPSMR)

When an FCC is configured for HDLC mode, the FPSMR is used as the HDLC mode register, shown in Figure 38-3.

	0–3	4	5	6–8	9	10–15
Field	NOF	FSE	MFF	—	TS	—
Reset	0000_0000_0000_0000					
R/W	R/W					
Offset	0x0x9_1304 (FPSMR1), 0x0x9_1324 (FPSMR2)					

	16	17–23	24	25	26–31
Field	NBL	—	CRC		—
Reset	0000_0000_0000_0000				
R/W	R/W				
Offset	0x9_1306 (FPSMR1), 0x9_1326 (FPSMR2)				

Figure 38-3. HDLC Mode Register (FPSMR)

The FPSMR fields are described in Table 38-6.

Table 38-6. FPSMR Field Descriptions [1]

Bits	Name	Description
0–3	NOF	Number of flags. Minimum number of flags between or before frames (0–15 flags). If NOF = 0000, no flags are inserted between the frames. Thus, for back-to-back frames, the closing flag of one frame is immediately followed by the opening flag of the next frame.
4	FSE	Flag sharing enable. This bit is valid only if GFMR[RTSM] is set. 0 Normal operation 1 If NOF = 0000, a single shared flag is transmitted between back-to-back frames. Other values of NOF are decremented by 1 when FSE is set. This is useful in signaling system #7 applications.
5	MFF	Multiple frames in FIFO. Setting MFF applies only when in $\overline{\text{RTS}}$ mode (GFMR*x*[RTSM] = 1). 0 Normal operation. The transmit FIFO buffer must never contain more than one HDLC frame. The $\overline{\text{CTS}}$ lost status is reported accurately on a per-frame basis. The receiver is not affected by this bit. 1 The transmit FIFO buffer can contain multiple frames, but lost $\overline{\text{CTS}}$ is not guaranteed to be reported on the exact buffer/frame it occurred on. This option, however, can improve the performance of HDLC transmissions for small back-to-back frames or if the user prefers to strongly limit the number of flags sent between frames. MFF does not affect the receiver. Refer to note 1 at the end of this table.
7–8	—	Reserved, should be cleared.
9	TS	Time stamp 0 Normal operation. 1 A 32-bit time stamp is added at the beginning of the receive BD data buffer, thus the buffer pointer must be (32-byte aligned - 4). The BD's data length does not include the time stamp. See Section 21.2.7, "RISC Time-Stamp Control Register (RTSCR)."
10–15	—	Reserved, should be cleared.

Table 38-6. FPSMR Field Descriptions (continued)[1]

Bits	Name	Description
16	NBL	Nibble mode enable 0 Nibble mode disabled (1 bit of data per clock). Note that at the end of the frame (after the closing flag), $\overline{\text{RTS}}$ negates immediately after the active edge of TCLK. 1 Nibble mode enabled (4 bits of data per clock). The negation of the $\overline{\text{RTS}}$ output signal is not synchronized to the serial clock. The $\overline{\text{RTS}}$ is negated after the last nibble of the data and always before the next edge of the serial clock. Note that at the end of the frame (after the closing flag), $\overline{\text{RTS}}$ negates a maximum of 5 CPM clocks after the active edge of TCLK.
17–23	—	Reserved, should be cleared.
24–25	CRC	CRC selection 00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1 01 Reserved 10 32-bit CCITT-CRC (Ethernet and HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1 11 Reserved
26–31	—	Reserved, should be cleared.

[1] When operating an FCC in HDLC nibble mode with the multiframe per FIFO bit off (FPSMR[MFF] = 0), the CPM might lose synchronization with the FCC HDLC controller. As a result the HDLC controller will become stuck and stop transmission. Therefore in HDLC nibble mode, FPSMR[MFF] must be set or the FCC must alternatively operate in HDLC bit mode.

38.7 HDLC Receive Buffer Descriptor (RxBD)

The HDLC controller uses the RxBD to report on data received for each buffer. Figure 38-4 shows an example of the RxBD process.

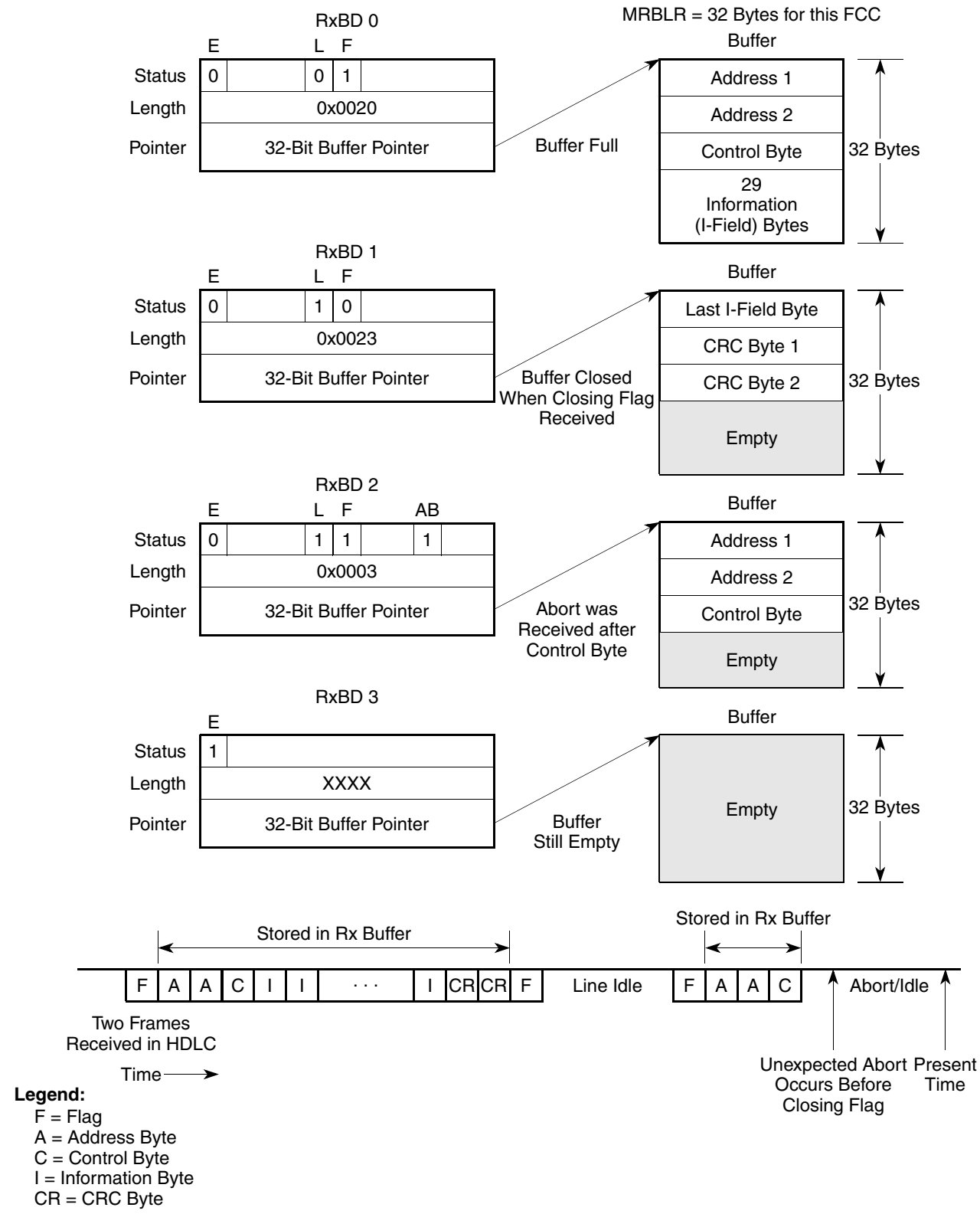


Figure 38-4. FCC HDLC Receiving Using RxBDs

Figure 38-5 shows the FCC HDLC RxBD.

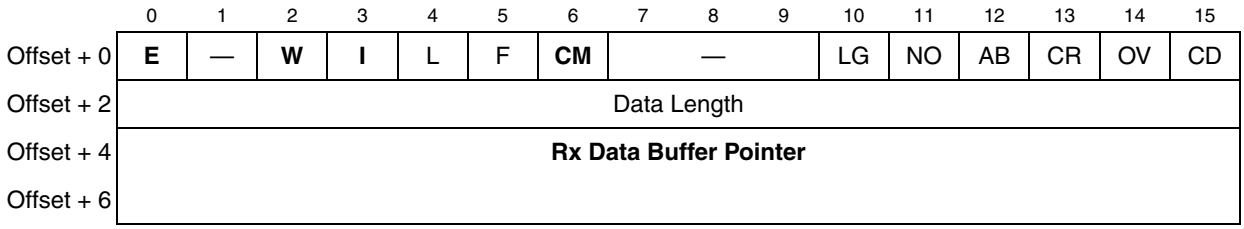


Figure 38-5. FCC HDLC Receive Buffer Descriptor (RxBD)

Table 38-7 describes RxBD fields.

Table 38-7. RxBD Field Descriptions

Bits	Name[1]	Description
0	**E**	Empty 0 The buffer is full with received data or data reception stopped because of an error. The core can read or write to any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer associated with this BD is empty. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (final BD in table) 0 Not the last BD in the RxBD table. 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is programmable and is determined only by the W bit and the overall space constraints of the dual-port RAM. The RxBD table must contain more than one BD in HDLC mode.
3	**I**	Interrupt 0 The RXB bit is not set after this buffer is used, but RXF operation remains unaffected. 1 FCCE[RXB] or FCCE[RXF] is set when the HDLC controller uses this buffer. These two bits can cause interrupts if they are enabled.
4	L	Last in frame. Set by the HDLC controller when this buffer is the last one in a frame. This implies the reception of a closing flag or reception of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 Not the last buffer in a frame. 1 Last buffer in a frame.
5	F	First in frame. Set by the HDLC controller when this buffer is the first in a frame. 0 Not the first buffer in a frame. 1 First buffer in a frame.
6	**CM**	Continuous mode 0 Normal operation. 1 The E bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be automatically overwritten the next time the CP accesses this BD. However, the E bit is cleared if an error occurs during reception, regardless of the CM bit.
7–9	—	Reserved, should be cleared.
10	LG	Rx frame length violation. A frame length greater than the maximum defined for this channel is recognized, and only the maximum-allowed number of bytes (MFLR) is written to the data buffer. This event is not reported until the RxBD is closed, the RXF bit is set, and the closing flag is received. The number of bytes received between flags is written to the data length field of this BD.

Table 38-7. RxBD Field Descriptions (continued)

Bits	Name[1]	Description
11	NO	Rx nonoctet-aligned frame. Set when a received frame contains a number of bits not divisible by eight.
12	AB	Rx abort sequence. At least seven consecutive 1s are received during frame reception.
13	CR	Rx CRC error. This frame contains a CRC error. Received CRC bytes are written to the receive buffer.
14	OV	Overrun. A receiver overrun occurs during frame reception.
15	CD	Carrier detect lost. $\overline{CD}$ has negated during frame reception. This bit is valid only for NMSI mode.

1 **Boldfaced** entries must be initialized by the user.

The RxBD status bits are written by the HDLC controller after receiving the associated data buffer.

The remaining RxBD parameters are as follows:

- Data length is the number of octets the CP writes into this BD's data buffer. It is written by the CP once the BD is closed. When this is the last BD in the frame (L = 1), this field contains the total number of frame octets, including 2 or 4 bytes for CRC. The memory allocated for this buffer should be no smaller than the MRBLR value.
- Rx data buffer pointer. The receive buffer pointer, which always points to the first location of the associated data buffer, resides in internal or external memory and must be divisible by 32 unless FPSMR[TS] = 1 (see Table 38-6).

38.8 HDLC Transmit Buffer Descriptor (TxBD)

Data is presented to the HDLC controller for transmission on an FCC channel by arranging it in buffers referenced by the channel TxBD table. The HDLC controller confirms transmission (or indicates errors) using the BDs to inform the core that the buffers have been serviced. Figure 38-6 shows the FCC HDLC TxBD.

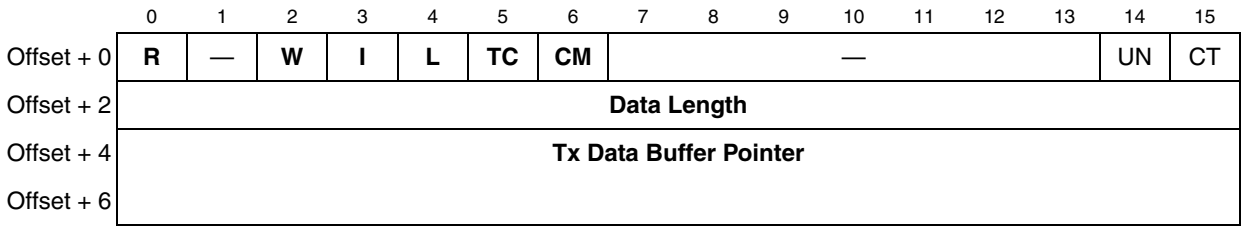


Figure 38-6. FCC HDLC Transmit Buffer Descriptor (TxBD)

Table 38-8 describes HDLC TxBD fields.

Table 38-8. HDLC TxBD Field Descriptions

Bits	Name[1]	Description
0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user can manipulate this BD or its associated buffer. The CP clears R after the buffer has been sent or an error occurs. 1 The buffer is ready to be sent. The transmission may have begun, but it has not completed. The user cannot set fields in this BD once R is set.
1	—	Reserved, should be cleared.
2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer has been used, the CP sends data from the first BD that TBASE points to in the table. The number of TxBDs in this table is determined only by the W bit and the overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 Either FCCE[TXB] or FCCE[TXE] is set when this buffer is serviced by the HDLC controller. These bits can cause interrupts if they are enabled.
4	**L**	Last 0 Not the last buffer in the frame. 1 Last buffer in the current frame.
5	**TC**	Tx CRC.Valid only when the L bit is set. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	**CM**	Continuous mode 0 Normal operation. 1 The R bit is not cleared by the CP after this BD is closed, allowing the buffer to be retransmitted automatically the next time the CP accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. The HDLC controller encounters a transmitter underrun condition while sending the buffer. The HDLC controller writes UN after sending the buffer.
15	CT	$\overline{\text{CTS}}$ lost. Set when $\overline{\text{CTS}}$ is lost during frame transmission in NMSI mode. If data from more than one buffer is in the FIFO buffer when this error occurs, CT is set in the currently open TxBD. The HDLC controller writes CT after sending the buffer.

[1] **Boldfaced** entries must be initialized by the user.

The TxBD status bits are written by the HDLC controller after sending the associated data buffer.

The remaining TxBD parameters are as follows:

- Data length is the number of bytes the HDLC controller should transmit from this data buffer; it is never modified by the CP. The value of this field should be greater than zero.
- Tx data buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, can be even or odd. The buffer can reside in internal or external memory. This value is never modified by the CP.

38.9 HDLC Event Register (FCCE)/Mask Register (FCCM)

The FCCE is used as the HDLC event register when the FCC operates as an HDLC controller. The FCCE reports events recognized by the HDLC channel and generates interrupts. On recognition of an event, the HDLC controller sets the corresponding FCCE bit. FCCE bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the CP clears the internal interrupt request.

Interrupts generated by the FCCE can be masked in the HDLC mask register (FCCM), which has the same bit format as FCCE. If an FCCM bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked.

Figure 38-7 represents the FCC/FCCM.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—								GRA	—		TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x0x9_1310 (FCCE1), 0x0x9_1330 (FCCE2), 0x0x9_1314 (FCCM1), 0x0x9_1334 (FCCM2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—						FLG	IDL	—							
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1312 (FCCE1), 0x9_1332 (FCCE2), 0x9_1316 (FCCM1), 0x9_1336 (FCCM2)															

Figure 38-7. HDLC Event Register (FCCE)/Mask Register (FCCM)

Table 38-9 describes FCCE/FCCM fields.

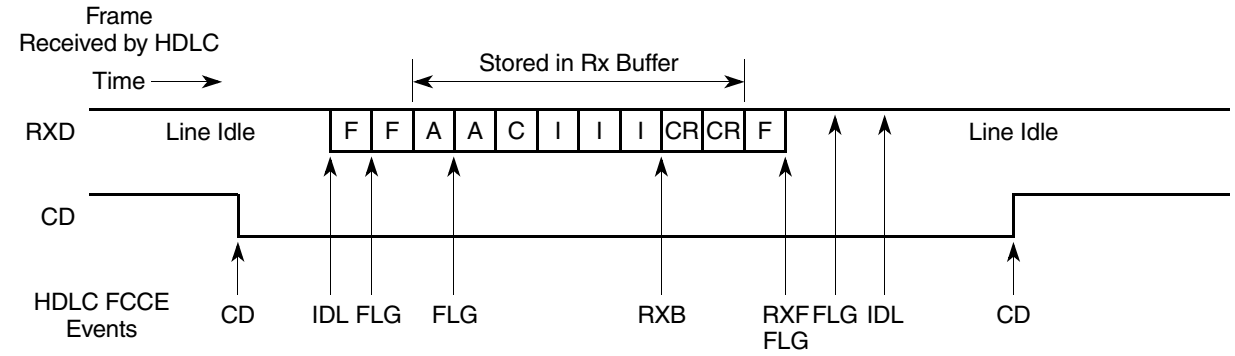
Table 38-9. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. GRA is set as soon as the transmitter finishes transmitting any frame that is in progress when the command was issued. It is set immediately if no frame is in progress when the command is issued.
9–10	—	Reserved, should be cleared.
11	TXE	Tx error. An error ($\overline{\text{CTS}}$ lost or underrun) occurs on the transmitter channel.
12	RXF	Rx frame. A complete frame is received on the HDLC channel. This bit is set no sooner than two clocks after receipt of the last bit of the closing flag.
13	BSY	Busy condition. A frame is received and discarded due to a lack of buffers.
14	TXB	Transmit buffer. Enabled by setting TxBD[I]. A buffer is sent on the HDLC channel. TXB is set no sooner than when the last bit of the closing flag begins its transmission if the buffer is the last one in the frame. Otherwise, TXB is set after the last byte of the buffer is written to the transmit FIFO buffer.

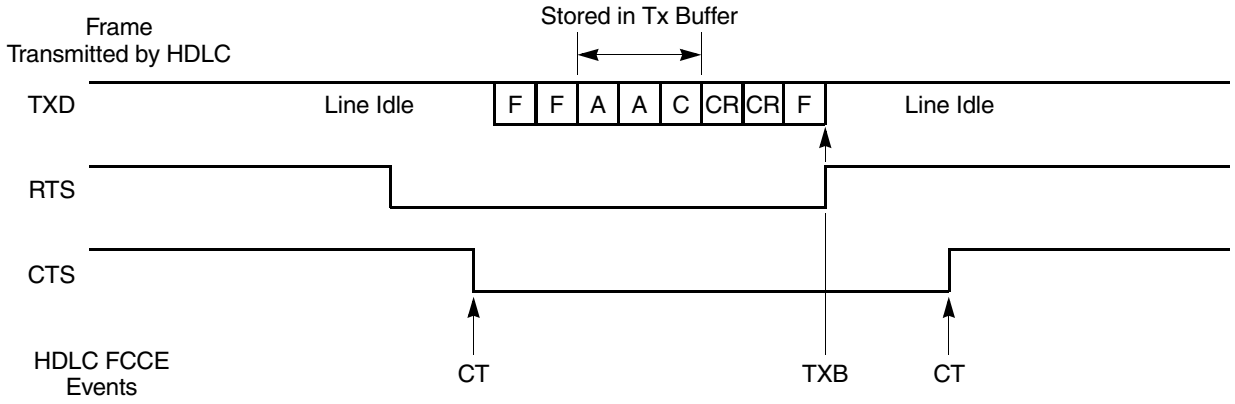
Table 38-9. FCCE/FCCM Field Descriptions (continued)

Bits	Name	Description
15	RXB	Receive buffer. When RXB = 1, a buffer for which the I bit is set in the corresponding BD was filled, regardless if the end of a frame was completed in it.
16–21	—	Reserved, should be cleared.
22	FLG	Flag status changed. The HDLC controller stops or starts receiving HDLC flags. The real-time status can be obtained in FCCS; see Section 38.10, “FCC Status Register (FCCS).”
23	IDL	Idle sequence status changed. A change in the status of the serial line is detected on the HDLC line. The real-time status can be read in FCCS; see Section 38.10, “FCC Status Register (FCCS).”
24–31	—	Reserved, should be cleared.

Figure 38-8 shows interrupts that can be generated in the HDLC protocol.



Notes:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.
3. The FLG interrupts show the beginning and end of flag reception.
4. The FLG interrupt at the end of the frame may precede the RXF interrupt due to receive FIFO latency.
5. The CD event must be programmed in the parallel I/O port, not in the FCC itself.
6. F = flag, A = address byte, C = control byte, I = information byte, and CR = CRC byte



Notes:
1. TXB event shown assumes all three bytes were put into a single buffer.
2. Example shows one additional opening flag. This is programmable.
3. The CT event must be programmed in the parallel I/O port, not in the FCC itself.

Figure 38-8. HDLC Interrupt Event Example

38.10 FCC Status Register (FCCS)

The FCCS register, shown in Figure 38-9, allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ signals are part of the parallel I/O port; see Chapter 45, "Parallel I/O Ports."

	0	1	2	3	4	5	6	7
Field	—					FG	—	ID
Reset	0000_0000							
R/W	R							
Offset	0x0x9_1318 (FCCS1), 0x0x9_1338 (FCCS2)							

Figure 38-9. FCC Status Register (FCCS)

Table 38-10 describes FCCS bits.

Table 38-10. FCCS Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	FG	Flags. While FG is cleared, each time a new bit is received the most recently received 8 bits are examined to see if a flag is present. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once FG is set, it remains set at least 8 bit times while the next 8 bits of input data are examined. If another flag occurs, FG stays set for at least another eight bits. Otherwise, FG is cleared and the search begins again. 0 HDLC flags are not currently being received. 1 HDLC flags are currently being received.
6	—	Reserved, should be cleared.
7	ID	Idle status. ID is set when the RXD signal is a logic one for 15 or more consecutive bit times; it is cleared after a logic zero is received. 0 The line is busy. 1 The line is idle.

Chapter 39
FCC Transparent Controller

The FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter. Transparent mode provides a clear channel on which the FCC performs no bit-level manipulation—implementing higher-level protocols would require software. Transparent mode is also referred to as a totally transparent or promiscuous operation.

Basic applications for an FCC in transparent mode include the following:

- For data, such as voice, moving serially without the need for protocol processing
- For board-level applications, such as chip-to-chip communications, requiring a serial-to-parallel and parallel-to-serial conversion
- For applications requiring the switching of data paths without altering the protocol encoding itself, such as a multiplexer in which data from a high-speed TDM serial stream is divided into multiple low-speed data streams

An FCC transmitter and receiver can be programmed in transparent mode independently. Setting GFMR*x*[TTx] enables the transparent transmitter; setting GFMR*x*[TRx] enables the transparent receiver. Both bits must be set for full-duplex transparent operation. If only one bit is set, the other half of the FCC operates with the protocol programmed in GFMR*x*[MODE]. This allows loopback modes to transfer data from one memory location to another (using DMA) while the data is converted to a specific serial format. However, the Ethernet and ATM controllers cannot be split in this way. See Section 37.2, “General FCC Mode Registers (GFMRx).”

The FCC in transparent mode can work with the TSA or NMSI and support modem lines using the general-purpose I/O signals. The data can be transmitted and received with msb or lsb first in each octet. The FCC consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to the other FCCs. Each clock can be supplied from the internal BRG bank or external signals.

39.1 Features

The following is a list of the transparent controller’s important features:

- Flexible data buffers
- Automatic SYNC detection on receive
  - 16-bit pattern
  - 8-bit pattern
  - Automatic sync (always synchronized)
  - External sync signal support
- CRCs can optionally be transmitted and received

- Reverse data mode
- Another protocol can be performed on the FCC's other half (transmitter or receiver) during transparent mode
- External BD table

39.2 Transparent Channel Operation

The transparent transmitter and receiver operates in the same way as the HDLC controller of the FCC (see Chapter 38, "FCC HDLC Controller,") except in the following ways:

1. The FPSMR does not affect the transparent controller, only the GFMR does.
2. In Table 38-1, MFLR, HMASK, RFTHR, and RFCNT must be cleared for proper operation of the transparent receiver.
3. Transmitter synchronization has to be achieved using $\overline{\text{CTS}}$ before the transmitter begins sending; see Section 39.3, "Achieving Synchronization in Transparent Mode."

39.3 Achieving Synchronization in Transparent Mode

Once the FCC transmitter is enabled for transparent operation in the GFMR, the TxBD is prepared for the FCC, and the transmit FIFO is preloaded by the SDMA channel, transmit synchronization must be established before data can be sent.

Similarly, once the FCC receiver is enabled for transparent operation in the GFMR and the RxBD is made empty for the FCC, receive synchronization must occur before data can be received. The synchronization process gives the user bit-level control of when the transmission and reception begins. The methods for this are as follows:

- An in-line synchronization pattern
- External synchronization signals
- Automatic sync

39.3.1 In-Line Synchronization Pattern

The transparent channel can be programmed to transmit and receive a synchronization pattern if GFMR[SYNL] ≠ 0; see Section 37.2, "General FCC Mode Registers (GFMRx)." The pattern is defined in the FDSR; see Section 37.5, "FCC Data Synchronization Registers (FDSRx)." GFMR[SYNL] defines the SYNC pattern length. The synchronization pattern is shown in Figure 39-1.

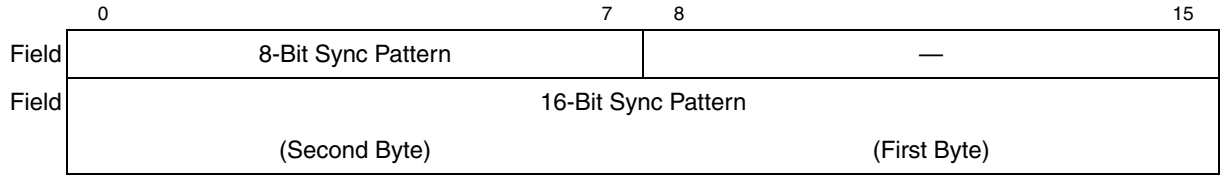


Figure 39-1. In-Line Synchronization Pattern

The receiver synchronizes on the synchronization pattern located in the FDSR. For instance, if an 8-bit SYNC is selected, reception begins as soon as these eight bits are received, beginning with the first bit

following the 8-bit SYNC. This effectively links the transmitter synchronization to the receiver synchronization.

39.3.2 External Synchronization Signals

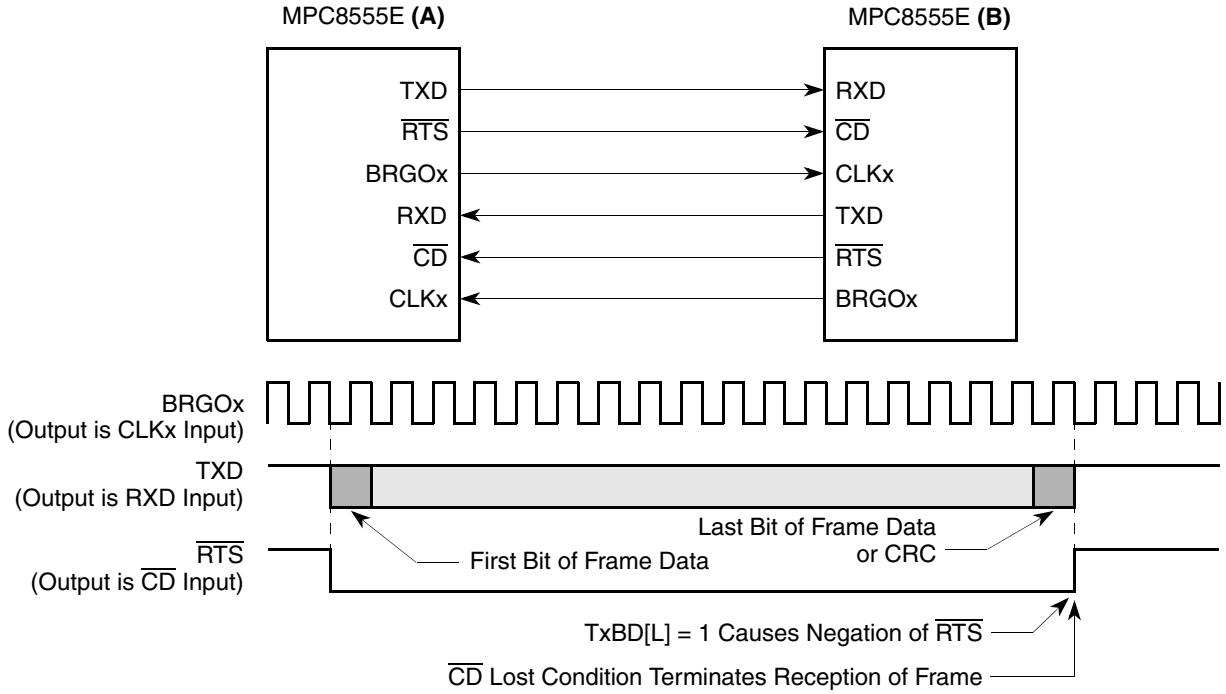
If GFMR[SYNL] = 00, an external signal is used to begin the sequence. $\overline{\text{CTS}}$ is used for the transmitter and $\overline{\text{CD}}$ is used for the receiver; these signals share the following sampling options:

- The pulse/envelope option determines whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ need to be asserted only once to begin reception/transmission or whether they must be asserted and stay that way for the duration of the transparent frame. This option is controlled by the CDP and CTSP bits of the GFMR. If the user expects a continuous stream of data without interruption, the pulse option should be used. However, if the user needs to identify frames of transparent data, the envelope mode of the these signals should be used. Note that the first bit of a frame is transmitted as zero every time $\overline{\text{RTS}}$ is asserted before $\overline{\text{CTS}}$ is asserted (GFMR[CTSS] = 1); subsequent data bits are sent accurately. Similarly, if $\overline{\text{CTS}}$ is in pulse mode (GFMR[CTSP] = 1), only the first frame is affected. If $\overline{\text{CTS}}$ is not in pulse mode (GFMR[CTSP] = 0), every frame is affected separately. Note that if NRZI encoding is used (GFMR[TENC] = 01), $\overline{\text{RTS}}$ must be asserted before $\overline{\text{CTS}}$, or else the first bit of the frame might be corrupted.
- The sampling option determines the delay between $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ being asserted and the resulting action by the FCC. These signals can be assumed to be asynchronous to the data and then internally synchronized by the FCC, or they can be assumed to be synchronous to the data giving faster operation. This option allows the $\overline{\text{RTS}}$ of one FCC to be connected to the $\overline{\text{CD}}$ of another FCC (on another MPC8555E) and to have the data synchronized and bit aligned. It is also an option to link the transmitter synchronization to the receiver synchronization.

When working with the FCC receiver in envelope mode, $\overline{\text{RTS}}$ should be asserted for at least 3 clock cycles between frames. Otherwise, the receiver cannot recognize the start of a new frame. Diagrams for the pulse/envelope and sampling options are in Section 37.12, “FCC Timing Control.”

39.3.3 Transparent Synchronization Example

Figure 39-2 shows an example of synchronization using external signals.



NOTES:

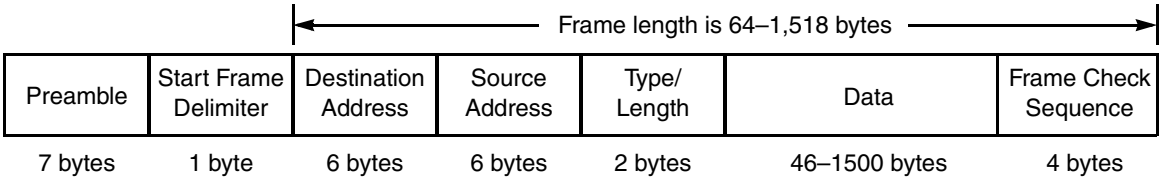
1. Each MPC8555E generates its own transmit clocks. If the transmit and receive clocks are the same, one can generate transmit and receive clocks for the other MPC8555E. For example, CLKx on MPC8555E (B) could be used to clock the transmitter and receiver.
2. $\overline{\text{CTS}}$ should be configured as always asserted in the parallel I/O port or connected to ground externally.
3. The required GSMR configurations are DIAG = 00, CTSS = 1, CTSP is a don't care, CDS = 1, CDP = 0, TTX = 1, and TRX = 1. REVD and TCRC are application-dependent.
4. The transparent frame contains a CRC if TxBD[TC] is set.

Figure 39-2. Sending Transparent Frames Between MPC8555Es

MPC8555E(A) and MPC8555E(B) exchange transparent frames and synchronize each other using $\overline{\text{RTS}}$ and $\overline{\text{CD}}$. However, $\overline{\text{CTS}}$ is not required because transmission begins at any time. Thus, $\overline{\text{RTS}}$ is connected directly to the other MPC8555E $\overline{\text{CD}}$. GFMR[SYNL] is not used and transmission and reception from each MPC8555E are independent.

Chapter 40
CPM Fast Ethernet Controller

The Ethernet IEEE 802.3 standard protocol is a widely-used LAN based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 standard protocols are similar and can coexist on the same LAN, both are referred to as Ethernet in this manual, unless otherwise noted. Ethernet/IEEE 802.3 standard frames are based on the frame structure shown in Figure 40-1.



Note: The lsb of each octet is transmitted first.

Figure 40-1. Ethernet Frame Structure

The elements of an Ethernet frame are as follows:

- 7-byte preamble of alternating ones and zeros
- Start frame delimiter (SFD)—Signifies the beginning of the frame
- 48-bit destination address
- 48-bit source address. Original versions of the IEEE 802.3 standard allowed 16-bit addressing, which has never been used widely.
- Ethernet type field/IEEE 802.3 standard length field. The type field signifies the protocol used in the rest of the frame, such as TCP/IP; the length field specifies the length of the data portion of the frame. For Ethernet and IEEE 802.3 standard frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This requirement limits the length of the data portion of the frame to 1,500 bytes and, therefore, the total frame length to 1,518 bytes.
- Data
- Four-bytes frame-check sequence (FCS), which is the standard, 32-bit CCITT-CRC polynomial used in many protocols

When a station needs to transmit, it waits until the LAN becomes silent for a specified period (interframe gap). When a station starts sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a frame. The station then waits a random time period (backoff) before attempting to send again. When the backoff completes, the station waits for silence on the LAN and then begins retransmission on the LAN. This process is called a retry. If the frame is not successfully sent within 15 retries, an error is indicated.

10-Mbps Ethernet basic timing specifications follow:

- Transmits at 0.8 µs per byte
- The preamble plus start frame delimiter is sent in 6.4 µs
- The minimum interframe gap is 9.6 µs
- The slot time is 51.2 µs

100-Mbps Ethernet basic timing specifications follow:

- Transmits at 0.08 µs per byte
- The preamble plus start frame delimiter is sent in 0.64 µs
- The minimum interframe gap is 0.96 µs
- The slot time is 5.12 µs

40.1 Fast Ethernet on the MPC8555E

When a general FCC mode register (GFMR*x*[MODE]) selects Ethernet protocol, that FCC performs the full set of IEEE 802.3 standard/Ethernet CSMA/CD media access control (MAC) and channel interface functions. Figure 40-2 shows a block diagram of the FCC Ethernet control logic.

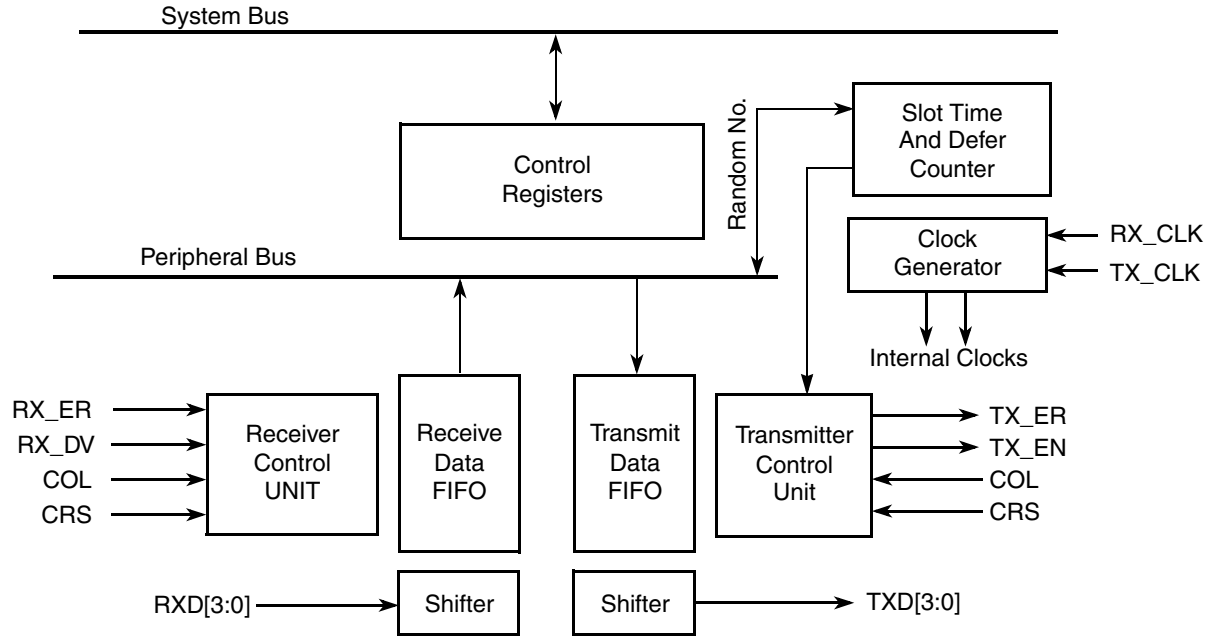


Figure 40-2. Ethernet Block Diagram

40.2 Features

The following is a list of Fast Ethernet key features:

- Support for Fast Ethernet through the MII (media-independent interface) and the RMII (reduced media-independent interface)
- Performs MAC (media access control) layer functions of Fast Ethernet and IEEE 802.3x standard

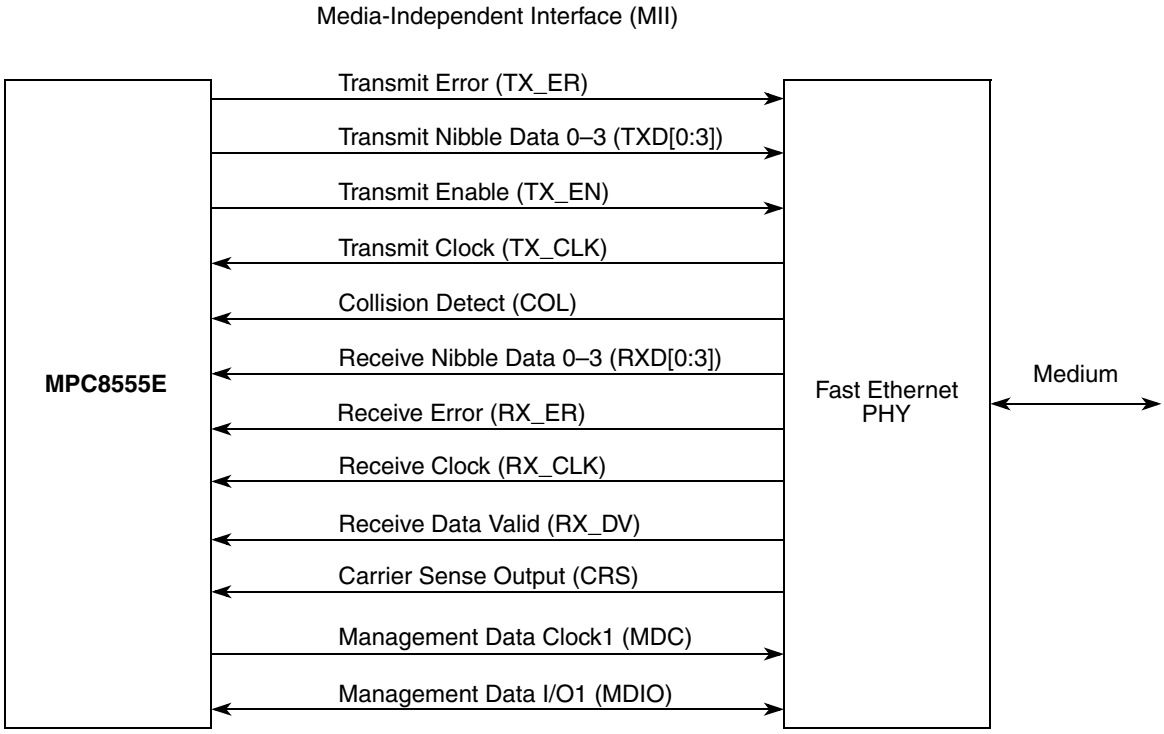
- Performs framing functions
  - Preamble generation and stripping
  - Destination address checking
  - CRC generation and checking
  - Automatic padding of short frames on transmit
  - Framing error (dribbling bits) handling
- Full collision support
  - Enforces the collision (jamming and TX_ER assertion)
  - Truncated binary exponential backoff algorithm for random wait
  - Two nonaggressive backoff modes
  - Automatic frame retransmission (until retry limit is reached)
  - Automatic discard of incoming collided frames
  - Delay transmission of new frames for specified interframe gap
- Bit rates up to 100 Mbps
- Receives back-to-back frames
- Detection of receive frames that are too long
- Multibuffer data structure
- Supports 48-bit addresses in three modes
  - Physical. One 48-bit address recognized or 64-bin hash table for physical addresses
  - Logical. 64-bin group address hash table plus broadcast address checking
  - Promiscuous. Receives all frames regardless of address (a CAM can be used for address filtering)
- External CAM support on system bus interfaces
- Special RMON counters for monitoring network statistics
- Transmitter network management and diagnostics
  - Lost carrier sense
  - Underrun
  - Number of collisions exceeded the maximum allowed
  - Number of retries per frame
  - Deferred frame indication
  - Late collision
- Receiver network management and diagnostics
  - CRC error indication
  - Nonoctet alignment error
  - Frame too short
  - Frame too long

  - Overrun
  - Busy (out of buffers)
- Error counters
  - Discarded frames (out of buffers or overrun occurred)
  - CRC errors
  - Alignment errors
- Internal and external loopback mode
- Supports Fast Ethernet in duplex mode
- Supports pause flow control frames
- Support of out-of-sequence transmit queue (for flow-control frames)
- External buffer descriptors (BDs)

40.3 Connecting the MPC8555E to Fast Ethernet

40.3.1 Connecting the MPC8555E to Ethernet (MII)

Figure 40-3 shows the basic components of the media-independent interface (MII) and the signals required to make the Fast Ethernet connection between the MPC8555E and a PHY.



[1] The management signals (MDC and MDIO) can be common to all of the Fast Ethernet connections in the system, assuming that each PHY has a different management address. Use parallel I/O port pins to implement MDC and MDIO. (The I^2C controller cannot be used for this function.)

Figure 40-3. Connecting the MPC8555E to Ethernet

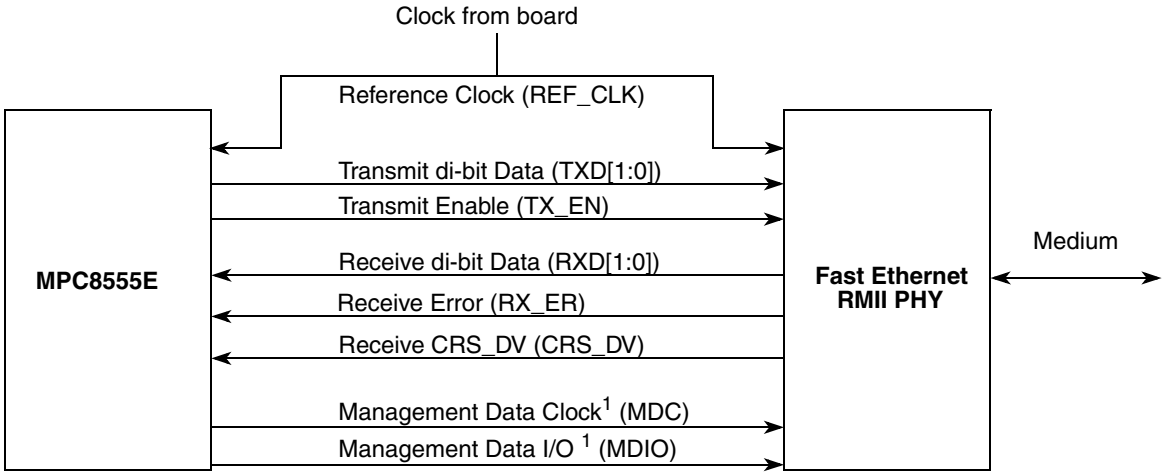
Each FCC has 18 signals, defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY. The two management signals (MDC and MDIO) required by the MII should be implemented separately using the parallel I/O.

The MPC8555E has additional signals for interfacing with an optional external content-addressable memory (CAM), which are described in Section 40.7, "CAM Interface."

The MPC8555E uses the SDMA channels to store every byte received after the start frame delimiter into system memory. On transmit, the user provides the destination address, source address, type/length field, and transmit data. To meet minimum frame requirements, MPC8555E automatically pads frames with fewer than 64 bytes in the data field. The MPC8555E also appends the FCS to the frame.

40.3.2 Connecting the MPC8555E to Ethernet (RMII)

Figure 40-4 shows the basic components of the reduced media-independent interface (RMII) and the signals required for the fast Ethernet connection between the MPC8555E and a PHY. The MDC/MDIO management interface is the same as in MII. The RMII reference clock (REF_CLK) is distributed over the FCC transmit clock. In RMII mode receive clock is not used.



[1]The management signals (MDC and MDIO) can be common to all of the fast Ethernet connections in the system, assuming that each PHY has a different management address. Use parallel I/O port pins to implement MDC and MDIO. (The I^2C controller cannot be used for this function.)

Figure 40-4. Connecting the MPC8555E to Ethernet (RMII)

40.4 Ethernet Channel Frame Transmission

The Ethernet transmitter requires almost no core intervention. When the core enables the transmitter, the Ethernet controller polls the first TxBD in the FCC's TxBD table every 256 serial clocks. If the user has a frame ready to transmit, setting FTODR[TOD] eliminates waiting for the next poll. When there is a frame to transmit, the Ethernet controller begins fetching the data from the data buffer and asserts TX_EN. The preamble sequence, start frame delimiter, and frame information are sent in that order; see Figure 40-1. In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap 28 serial clocks (112 bit-time period) regardless of CRS assertion.

There is one internal buffer for out-of-sequence flow control frames (in full-duplex Fast Ethernet). When the Fast Ethernet controller is between frames, this buffer is polled if flow control is enabled. This buffer must contain the whole frame.

However, in half-duplex mode, the controller defers transmission if the line is busy (CRS asserted). Before transmitting, the controller waits for carrier sense to become inactive, at which point the controller determines if CRS remains negated for 16 serial clocks. If so, the transmission begins after an additional 8 serial clocks (96 bit-times after CRS originally became negated). In the fast ethernet transmitter, if CRS is asserted and then negated within 10 clocks after TXEN is negated, the next frame is not deferred and a defer indication is asserted.

If a collision occurs during the transmit frame, the Ethernet controller follows a specified backoff procedure and tries to retransmit the frame until the retry limit is reached. The Ethernet controller stores at least the first 64 bytes of data of the transmit frame in the FCC FIFO, so that the data does not have to be retrieved from system memory in case of a collision. This improves bus usage and latency if the backoff timer output requires an immediate retransmission.

When the end of the current buffer is reached and TxBD[L] = 1, the FCS (32-bit CRC) bytes are appended (if TxBD[TC] = 1), and TX_EN is negated. This notifies the PHY of the need to generate the illegal Manchester encoding that signifies the end of an Ethernet frame. Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. When the end of the current buffer is reached and TxBD[L] = 0 (a frame is comprised of multiple buffers), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD] = 1, the Ethernet controller pads short frames to the value of the minimum frame length register (MINFLR), described in Table 40-2.

To rearrange the transmit queue before the CP finishes sending all frames, issue a GRACEFUL STOP TRANSMIT command. This can be useful for transmitting expedited data ahead of previously linked buffers or for error situations. When the GRACEFUL STOP TRANSMIT command is issued, the Ethernet controller stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with a collision. When the Ethernet controller is given the RESTART TRANSMIT command, it resumes transmission. The Ethernet controller sends bytes least-significant nibble first.

40.5 Ethernet Channel Frame Reception

The Ethernet receiver is designed to work with almost no core intervention and can perform address recognition, CRC checking, short frame checking, maximum DMA transfer checking, and maximum frame-length checking.

When the core enables the Ethernet receiver, it enters hunt mode when RX_DV is asserted as long as COL remains negated (full-duplex mode ignores COL). In hunt mode, as data is shifted into the receive shift register four bits at a time, the contents of the register are compared to the contents of the SYN2 field in the FCC's data synchronization register (FDSR). When the registers match, the hunt mode is terminated and character assembly begins.

When the receiver detects the first bytes of a frame, the Ethernet controller performs address recognition functions on the frame; see Section 40.12, “Ethernet Address Recognition.” The receiver can receive physical (individual), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal address recognition algorithm is complete, bus usage is not wasted on frames not addressed to this station. The receiver can also operate with an external CAM, in which case frame reception continues normally, unless the CAM specifically signals the frame to be rejected. See Section 40.7, “CAM Interface.”

If an address is recognized, the Ethernet controller fetches the next RxBD and, if it is empty, starts transferring the incoming frame to the RxBD’s associated data buffer.

In half-duplex mode, if a collision is detected during the frame, the RxBDs associated with this frame are reused. Thus, no collision frames are presented to the user except late collisions, which indicate serious LAN problems. When the buffer has been filled, the Ethernet controller clears RxBD[E] and generates an interrupt if RxBD[I] is set. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table; if it is empty, it continues receiving the rest of the frame.

The RxBD length is determined by MRBLR in the parameter RAM. The user should program MRBLR to be at least 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. When the frame ends, the receive CRC field is checked and written to the data buffer. The data length written to the last BD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

If an external CAM is used (FPSMR*x*[CAM] = 1), the Ethernet controller adds the 2 lower bytes of the CAM output at the end of each frame. Note that the data length does not include these 2 bytes; that is, the extra 2 bytes could push the buffer length past MRBLR.

When the receive frame is complete, the Ethernet controller sets RxBD[L], writes the other frame status bits into the RxBD, and clears RxBD[E]. The Ethernet controller next generates a maskable interrupt, indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame. The Ethernet controller receives serial data least-significant nibble first.

40.6 Flow Control

Because collisions cannot occur in full-duplex mode, Fast Ethernet can operate at the maximum rate. When the rate becomes too fast for a station’s receiver, the station’s transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 40-1 shows the flow-control frame structure.

Table 40-1. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble		
1	SFD		Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames

Table 40-1. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
6	Source address		
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter	Up to 0xFFFE	Pause period measured in slot times, most-significant octet first with a two time-slot resolution. **Note:** Because the pause period has a resolution of two time slots, the value programmed in this field is rounded up to the nearest even number before being used, as follows: MAC Parameter Value — Pause Period 0 — none 1 or 2 — 2 x slot time 3 or 4 — 4 x slot time … — …
42	Reserved	—	
4	FCS		Frame check sequence (CRC)

When flow-control mode is enabled (FPSMR*x*[FCE]) and the receiver identifies a pause-flow control frame sent to individual or broadcast addresses, transmission stops for the time specified in the control frame. During this pause, only the out-of-sequence frame is sent. Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

40.7 CAM Interface

The MPC8555E internal address recognition logic can be used in combination with an external CAM. When using a CAM, the FCC must be in promiscuous mode (FPSMR*x*[PRO] = 1). See Section 40.12, “Ethernet Address Recognition.”

The Ethernet controller writes two 32-bit accesses to the CAM and then reads the result in a 32-bit access. If the high bit of the result is set, the frame is rejected; otherwise, the lower 16 bits are attached to the end of the frame.

When an external CAM is used for address filtering, users can choose to either discard rejected frames (FPSMR[ECM] = 0) or receive rejected frames and signal the CAM miss in the RxBD (FPSMR[ECM] = 1).

NOTE

The bus atomicity mechanism for CAM accesses may not function correctly when the CPM performs a DMA access to an external CAM device. This only impacts systems in which multiple CPMs will access the CAM.

40.8 Ethernet Parameter RAM

For Ethernet mode, the protocol-specific area of the FCC parameter RAM is mapped as in Table 40-2.

Table 40-2. Ethernet-Specific Parameter RAM

Offset[1]	Name[2]	Width	Description
0x3C	STAT_BUF	Word	Buffer of internal usage
0x40	**CAM_PTR**	Word	CAM address. For FCC Fast Ethernet operation the CAM should be located on the same bus as the data buffers.
0x44	**C_MASK**	Word	Constant MASK for CRC (initialize to 0xDEBB_20E3). For the 32-bit CRC-CCITT.
0x48	**C_PRES**	Word	Preset CRC (initialize to 0xFFFF_FFFF). For the 32-bit CRC-CCITT.
0x4C	**CRCEC**[3]	Word	CRC error counter. Counts each received frame with a CRC error. Does not count frames not addressed to the station, frames received in the out-of-buffers condition, frames with overrun errors, or frames with alignment errors.
0x50	**ALEC** [3]	Word	Alignment error counter. Counts frames received with dribbling bits. Does not count frames not addressed to the station, frames received in the out-of-buffers condition, or frames with overrun errors.
0x54	**DISFC** [3]	Word	Discard frame counter. Incremented for discarded frames because of an out-of-buffers condition or overrun error. The CRC need not be correct for this counter to be incremented.
0x58	**RET_LIM**	Hword	Retry limit (typically 15 decimal). Number of retries that should be made to send a frame. If the frame is not sent after this limit is reached, an interrupt can be generated.
0x5A	RET_CNT	Hword	Retry limit counter. Temporary decrementer used to count retries made.
0x5C	**P_PER**	Hword	Persistence. Allows the Ethernet controller to be less persistent after a collision. Normally cleared, P_PER can be from 0 to 9 (9 = least persistent). The value is added to the retry count in the backoff algorithm to reduce the chance of transmission on the next time-slot. Using a less persistent backoff algorithm increases throughput in a congested Ethernet LAN by reducing the chance of collisions. FPSMR[SBT] can also reduce persistence of the Ethernet controller. The Ethernet/802.3 standard specifications permit the use of P_PER.
0x5E	BOFF_CNT	Hword	Backoff counter
0x60	**GADDR_H**	Word	Group address filters high and low are used in the hash table function of the group addressing mode. The user may write zeros to these values after reset and before the Ethernet channel is enabled to disable all group hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table. See Section 40.13, “Hash Table Algorithm.”
0x64	**GADDR_L**	Word	
0x68	**TFCSTAT**	Hword	Out-of-sequence TxBD. Includes the status/control, data length, and buffer pointer fields in the same format as a regular TxBD. Useful for sending flow control frames. This area’s TxBD[R] is always checked between frames, regardless of FPSMR*x*[FCE]. If it is not ready, a regular frame is sent. The user must set TxBD[L] when preparing this BD. If TxBD[I] is set, a TXC event is generated after frame transmission. This area should be cleared when not in use.
0x6A	**TFCLEN**	Hword	
0x6C	**TFCPTR**	Word	
0x70	**MFLR**	Hword	Maximum frame length register (typically1518 decimal). If the Ethernet controller detects an incoming frame exceeding MFLR, it sets RxBD[LG] (frame too long) in the last RxBD, but does not discard the rest of the frame. The controller also reports the frame status and length of the received frame in the last RxBD. MFLR includes all in-frame bytes between the start frame delimiter and the end of the frame.

Table 40-2. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name[2]	Width	Description
0x72	**PADDR1_H**	Hword	The 48-bit individual address of this station. PADDR1_L is the lowest order half-word, and PADDR1_H is the highest order half-word.
0x74	**PADDR1_M**	Hword	
0x76	**PADDR1_L**	Hword	
0x78	IBD_CNT	Hword	Internal BD counter
0x7A	IBD_START	Hword	Internal BD start pointer
0x7C	IBD_END	Hword	Internal BD end pointer
0x7E	TX_LEN	Hword	Tx frame length counter
0x80	IBD_BASE	32 Bytes	Internal microcode usage
0xA0	**IADDR_H**	Word	Individual address filter high/low. Used in the hash table function of the individual addressing mode. The user can write zeros to these values after reset and before the Ethernet channel is enabled to disable all individual hash address recognition functions. Issuing a SET GROUP ADDRESS command enables the hash table. See Section 40.13, "Hash Table Algorithm."
0xA4	**IADDR_L**	Word	
0xA8	**MINFLR**	Hword	Minimum frame length register (typically 64 decimal). If the Ethernet receiver detects an incoming frame shorter than MINFLR, it discards that frame unless FPSMR[RSH] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The Ethernet transmitter pads frames that are too short (according to TxBD[PAD] and the PAD value in the parameter RAM). PADs are added to make the transmit frame MINFLR bytes.
0xAA	**TADDR_H**	Hword	Allows addition of addresses to the individual and group hashing tables. After an address is placed in TADDR, issue a SET GROUP ADDRESS command. TADDR_L is the lowest-order half-word; TADDR_H is the highest. A zero in the I/G bit indicates an individual address; 1 indicates a group address.
0xAC	**TADDR_M**	Hword	
0xAE	**TADDR_L**	Hword	
0xB0	**PAD_PTR**	Hword	Internal PAD pointer. This internal 32-byte aligned pointer points to a 32-byte buffer filled with pad characters. The pads may be any value, but all the bytes should be the same to assure padding with a specific character. If a specific padding character is not needed, PAD_PTR should equal the internal temporary data pointer TIPTR; see Section 37.8, "FCC Parameter RAM."
0xB2	—	Hword	Reserved, should be cleared.
0xB4	CF_RANGE	Hword	Control frame range. Internal usage
0xB6	MAX_B	Hword	Maximum BD byte count. Internal usage
0xB8	**MAXD1**	Hword	Max DMA1 length register (typically 1520 decimal). Lets the user prevent system bus writes after a frame exceeds a specified size. The MAXD1 value is valid only if an address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the user-defined value in MAXD1, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame (or until MFLR bytes have been received) and reports the frame status and length (including the discarded bytes) in the last RxBD. This value must be greater than 32.

Table 40-2. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name[2]	Width	Description
0xBA	**MAXD2**	Hword	Max DMA2 length register (typically 1520 decimal). Lets the user prevent system bus writes after a frame exceeds a specified size. The value of MAXD2 is valid in promiscuous mode when no address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the value in MAXD2, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame (or until MFLR bytes are received) and reports frame status and length (including the discarded bytes) in the last RxBD. In a monitor station, MAXD2 can be much less than MAXD1 to receive entire frames addressed to this station, but receive only the headers of all other frames.This value must be less than MAXD1.
0xBC	MAXD	Hword	Rx maximum DMA. Internal usage
0xBE	DMA_CNT	Hword	Rx DMA counter. Temporary down-counter used to track the frame length.
0xC0	**OCTC** [3]	Word	(RMON mode only) The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).
0xC4	**COLC** [3]	Word	(RMON mode only) The best estimate of the total number of collisions on this Ethernet segment.
0xC8	**BROC** [3]	Word	(RMON mode only) The total number of good packets received that were directed to the broadcast address. Note that this does not include multicast packets.
0xCC	**MULC** [3]	Word	(RMON mode only) The total number of good packets received that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address.
0xD0	**USPC** [3]	Word	(RMON mode only) The total number of packets received that were less than 64 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.
0xD4	**FRGC** [3]	Word	(RMON mode only) The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that it is entirely normal for etherStatsFragments to increment because it counts both runts (which are normal occurrences due to collisions) and noise hits.
0xD8	**OSPC** [3]	Word	(RMON mode only) The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.
0xDC	**JBRC** [3]	Word	(RMON mode only) The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets), and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that this definition of jabber is different than the definition in IEEE-802.3 standard, Section 8.2.1.5 (10BASE5) and Section 10.3.1.4 (10BASE2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.
0xE0	**P64C** [3]	Word	(RMON mode only) The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).
0xE4	**P65C** [3]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).
0xE8	**P128C** [3]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).
0xEC	**P256C** [3]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).

Table 40-2. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name[2]	Width	Description
0xF0	**P512C** [3]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).
0xF4	**P1024C** [3]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 1024 and 1518 octets long inclusive (excluding framing bits but including FCS octets).
0xF8	CAM_BUF	Word	Internal buffer for CAM result
0xFC	—	Word	Reserved, should be cleared.

1 Offset from FCC base: 0x8400 (FCC1) and 0x8500 (FCC2); see Section 21.4.2, "Parameter RAM."

2 **Boldfaced** entries must be initialized by the user.

3 32-bit (modulo 232) counters maintained by the CP; cleared by the user while the channel is disabled.

40.9 Programming Model

The core configures an FCC to operate as an Ethernet controller using GFMR[MODE]. The receive errors (collision, overrun, nonoctet-aligned frame, short frame, frame too long, and CRC error) are reported through the RxBD. The transmit errors (underrun, heartbeat, late collision, retransmission limit, and carrier sense lost) are reported through the TxBD.

The user should program the FDSR as described in Section 37.5, "FCC Data Synchronization Registers (FDSRx)," with FDSR[SYN2] = 0xD5 and FDSR[SYN1] = 0x55.

40.10 Ethernet Command Set

The transmit and receive commands are issued to the CPCR; see Section 21.3, "Command Set."

NOTE

Before resetting the CPM, configure TX_EN ($\overline{\text{RTS}}$) to be an input.

Transmit commands that apply to Ethernet are described in Table 40-3.

Table 40-3. Transmit Commands

Command	Description
STOP TRANSMIT	When used with the Ethernet controller, this command violates a specific behavior of an Ethernet/IEEE 802.3 standard station. It should not be used.
GRACEFUL STOP TRANSMIT	Used to smoothly stop transmission after the current frame finishes sending or undergoes a collision (immediately if there is no frame being sent). FCCE[GRA] is set once transmission stops. Then the Ethernet transmit parameters (including BDs) can be modified by the user. The TBPTR points to the next TxBD in the table. Transmission begins when the R bit of the next BD is set and the RESTART TRANSMIT command is issued. Note that if the GRACEFUL STOP TRANSMIT command is issued and the current transmit frame ends in a collision, the TBPTR points to the beginning of the collided frame with TxBD[R] still set (the frame looks as if it was never sent).

Table 40-3. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Enables transmission of characters on the transmit channel. It is expected by the Ethernet controller after a GRACEFUL STOP TRANSMIT command or transmitter error (underrun, retransmission limit reached, or late collision). The Ethernet controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all the transmit parameters in this serial channel parameter RAM to their reset state. This command should be issued only when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Receive commands that apply to Ethernet are described in Table 40-4.

Table 40-4. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel in the FCC mode register is enabled, the channel is in the receive enable mode and uses the first BD in the table. This command is generally used to force the Ethernet receiver to abort reception of the current frame and enter hunt mode. In hunt mode, the Ethernet controller continually scans the input data stream for a transition of carrier sense from inactive to active followed by a preamble sequence and the start frame delimiter. After receiving the command, the current receive buffer is closed, the error status flags and length field are cleared, RxBD[E] (the empty bit) is set, and the CRC calculation is reset. Further frame reception uses the current RxBD. Note that short frames pending in the internal FIFO may be lost.
INIT RX PARAMETERS	Initializes all the receive parameters in this serial channel parameter RAM to their reset state and should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the receive and transmit parameters.
SET GROUP ADDRESS	Used to set one of the 64 bits of the four individual/group address hash filter registers (IADDR[1–4] or GADDR[1–4]). The individual or group address (48 bits) to be added to the hash table should be written to TADDR_L, TADDR_M, and TADDR_H in the parameter RAM prior to executing this command. The CP checks the I/G bit in the address stored in TADDR to determine whether to use the individual hash table or the group hash table. A 0 in the I/G bit indicates an individual address; 1 indicates a group address. This command can be executed at any time, regardless of whether the Ethernet channel is enabled.

If an address from the hash table must be deleted, the Ethernet channel must be disabled, the hash table registers must be cleared, and the SET GROUP ADDRESS command must be executed for the remaining preferred addresses. This is required because the hash table might have mapped multiple addresses to the same hash table bit.

40.11 RMON Support

The Fast Ethernet controller can automatically gather network statistics required for RMON without the need to receive all addresses using promiscuous mode. Setting FPSMR*x*[MON] enables RMON support.

The RMON statistics and their corresponding counters in the parameter RAM are described in Table 40-5.

Table 40-5. RMON Statistics and Counters

Statistic	Description	Counter
etherStatsDropEvents	The total number of events in which packets were detected as dropped by the probe due to lack of resources. Note that this may not be the number of packets dropped; it is the number of times this condition is detected.	DISFC
etherStatsOctets	The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).	OCTC
etherStatsPkts	The total number of packets (including bad packets, broadcast packets, and multicast packets) received.	USPC + OSPC + FRGC + JBRC + P64C + P65C + P128C + P256C + P512C + P1024C
etherStatsBroadcastPkts	The total number of good packets received that were directed to the broadcast address. Note that this does not include multicast packets.	BROC
etherStatsMulticastPkts	The total number of good packets received that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address.	MULC
etherStatsCRCAlignErrors	The total number of packets received that had a length (excluding framing bits but including FCS octets) of between 64 and 1518 octets, inclusive, but had either an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error).	CRCEC + ALEC -FRGC -JBRC
etherStatsUndersizePkts	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and were otherwise well-formed.	USPC
etherStatsOversizePkts	The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.	OSPC
etherStatsFragments	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that it is entirely normal for etherStatsFragments to increment, because it counts both runts (which are normal occurrences due to collisions) and noise hits.	FRGC
etherStatsJabbers	The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that this definition of jabber is different than the definition in IEEE-802.3 standard, Section 8.2.1.5 (10BASE5) and Section 10.3.1.4 (10BASE2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.	JBRC
etherStatsCollisions	The best estimate of the total number of collisions on this Ethernet segment.	COLC

Table 40-5. RMON Statistics and Counters (continued)

Statistic	Description	Counter
etherStatsPkts64Octets	The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).	P64C
etherStatsPkts65to127Octets	The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).	P65C
etherStatsPkts128to255Octets	The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).	P128C
etherStatsPkts256to511Octets	The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).	P256C
etherStatsPkts512to1023Octets	The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).	P512C
etherStatsPkts1024to1518Octets	The total number of packets (including bad packets) received that were between 1024 and 1518 octets long inclusive (excluding framing bits but including FCS octets).	P1024C

40.12 Ethernet Address Recognition

The Ethernet controller can filter the received frames based on different addressing types—physical (individual), group (multicast), broadcast (all-ones group address), and promiscuous. The difference between an individual address and a group address is determined by the I/G bit in the destination address field.

Figure 40-5 is a flowchart for address recognition on received frames.

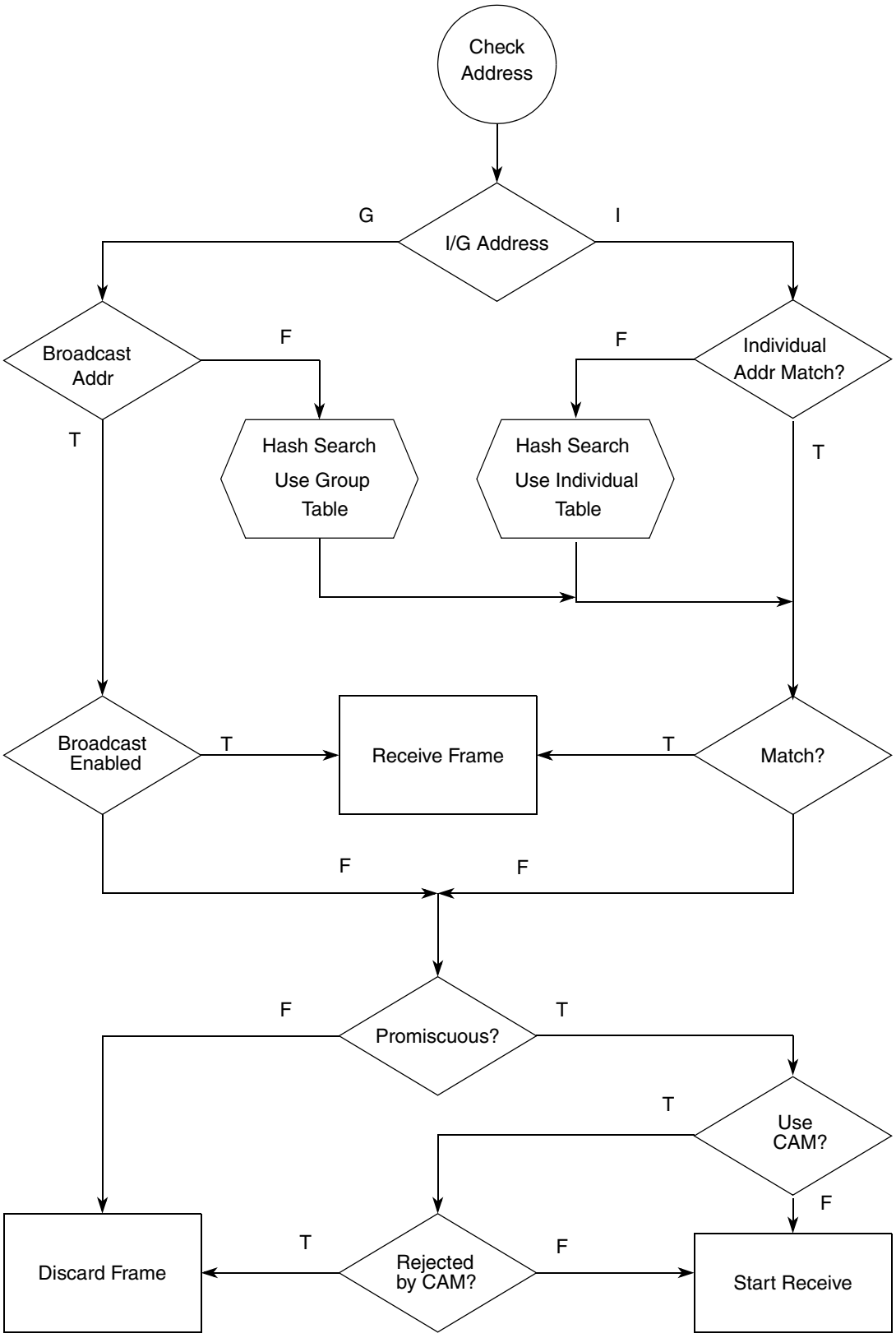


Figure 40-5. Ethernet Address Recognition Flowchart

In the physical type of address recognition, the Ethernet controller compares the destination address field of the received frame with the physical address that the user programs in the PADDR. If it fails, the controller performs address recognition on multiple individual addresses using the IADDR_H/L hash table. Since the controller always checks PADDR and the individual hash, for individual address the user must write zeros to the hash in order to avoid a hash match and ones to PADDR in order to avoid individual address match.

In the group type of address recognition, the Ethernet controller determines whether the group address is a broadcast address. If it is a broadcast and broadcast addresses are enabled, the frame is accepted. If the group address is not a broadcast address, the user can perform address recognition on multiple group addresses using the GADDR_H/L hash table. In promiscuous mode, the Ethernet controller receives all of the incoming frames regardless of their address when an external CAM is not used.

If an external CAM is used for address recognition (FPSMR[CAM] = 1), the user should select promiscuous mode; the frame can be rejected if there is no match in the CAM. If the on-chip address recognition functions detect a match, the external CAM is not accessed. Otherwise, the CPM issues a match transaction to the CAM using the bus on which the data buffers reside. (The data buffer bus is selected in FCR*x*[DTB]; see Section 37.8.1, “FCC Function Code Registers (FCRx).”)

40.13 Hash Table Algorithm

The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit address into one of 64 bins, which are represented by the 64 bits in GADDR_H/L or IADDR_H/L. When the SET GROUP ADDRESS command is executed, the Ethernet controller maps the selected 48-bit address in TADDR into one of the 64 bits. This is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and using 6 bits of the CRC-encoded result to generate a number between 1 and 64. Bit 26 of the CRC result selects which address filter registers are used in the hashing process—either GADDR_H/IADDR_H or GADDR_L/IADDR_L— and bits 27–31 of the CRC result select which bit is set.

The same process is used when the Ethernet controller receives a frame. If the CRC generator selects a bit that is set in the group/individual hash table, the frame is accepted; otherwise, it is rejected. The result is that if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. The core must further filter those that reach memory to determine if they contain one of the eight preferred addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if eight group and eight physical addresses are stored in their respective hash tables, 87.5% of all frames (not just group address frames) are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, with 128 addresses stored in a 64-bin hash table, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory. In such instances, an external CAM is advised if the extra bus use cannot be tolerated. See Section 40.7, “CAM Interface.”

NOTE

The hash tables cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. Thus, an external CAM must be used to implement this function.

40.14 Interpacket Gap Time

The minimum interpacket gap time for back-to-back transmission is 96 bit-times. The receiver receives back-to-back frames with this minimum spacing. In addition, after the backoff algorithm, the transmitter waits for carrier sense to be negated before retransmitting the frame. The retransmission begins 96 bit-times after carrier sense is negated if it stays negated for at least 64 bit-times. So if there is no change in the carrier sense indication during the first 64 bit-times (16 serial clocks), the retransmission begins 96 clocks after carrier sense is first negated

40.15 Handling Collisions

If a collision occurs during frame transmission, the Ethernet controller continues transmission for at least 32-bit times, transmitting a jam pattern of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the sequence ends.

If a collision occurs within 64 byte-times, the process is retried. The transmitter waits a random number of slot times. (A slot time is 512 bit-times.) If a collision occurs after 64 byte-times, no retransmission is performed, FCCE[TXE] is set, and the buffer is closed with a late-collision error indication in TxBD[LC]. If a collision occurs during frame reception, reception is stopped. This error is reported only in the RxBD if the frame is at least as long as the MINFLR or if FPSMR[RSH] = 1.

40.16 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both receive and transmit FIFO buffers are used and the FCC operates in full-duplex mode. Both internal and external loopback are configured using combinations of FPSMR[LPB] and GFMR[DIAG]. Because of the full-duplex nature of the loopback operation, the performance of the other FCCs is degraded.

Internal loopback disconnects the FCC from the SI. The receive data is connected to the transmit data. The transmitted data from the transmit FIFO is received immediately into the receive FIFO. There is no heartbeat check in this mode.

In external loopback operation, the Ethernet controller listens for data received from the PHY while it is sending.

40.17 Ethernet Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the FCC event register. Transmission errors are described in Table 40-6.

Table 40-6. Transmission Errors

Error	Response
Transmitter underrun	The controller sends 32 bits that ensure a CRC error, terminates buffer transmission, closes the buffer, sets TxBD[UN] and FCCE[TXE]. The controller resumes transmission after receiving the RESTART TRANSMIT command.
Carrier sense lost during frame transmission	If no collision is detected in the frame, the controller sets TxBD[CSL] and FCCE[TXE], and it continues the buffer transmission normally. No retries are performed as a result of this error.
Retransmission attempts limit expired	The controller terminates buffer transmission, closes the buffer, sets TxBD[RL] and FCCE[TXE]. Transmission resumes after receiving the RESTART TRANSMIT command.
Late collision	The controller terminates buffer transmission, closes the buffer, sets TxBD[LC] and FCCE[TXE]. The controller resumes transmission after receiving the RESTART TRANSMIT command. Note that late collision parameters are defined in FPSMR[LCW].

Reception errors are described in Table 40-7.

Table 40-7. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller writes the received data byte to the internal FIFO buffer over the previously received byte. The previous data byte and frame status are lost. The controller closes the buffer, sets RxBD[OV] and FCCE[RXF], and increments the discarded frame counter (DISFC). The receiver then enters hunt mode.
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets FCCE[BSY] and increments the discarded frame counter (DISFC).
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits when the receive frame terminates as nonoctet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame nonoctet aligned (RxBD[NO]) error is reported, FCCE[RXF] is set, and the alignment error counter (ALEC) in the parameter RAM is incremented. If there is no CRC error, no error is reported.
CRC error	When a CRC error occurs, the controller closes the buffer, and sets RxBD[CR] and FCCE[RXF]. Also, the CRC error counter (CRCEC) in the parameter RAM is incremented. After receiving a frame with a CRC error, the receiver enters hunt mode. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

40.18 Fast Ethernet Registers

The following sections describe registers used for configuring and operating the Fast Ethernet controller.

40.18.1 General FCC Expansion Mode Register (GFEMR)

The general FCC expansion mode register (GFEMR) defines the expansion modes. It should be programmed according to the protocol used.

	0	1	2	3	7
Field	TIREM	LPB	CLK	—	
Reset	0000_0000				
R/W	R/W				
Offset	0x9_1390 (GFEMR1), 0x9_13B0(GFEMR2)				

Figure 40-6. General FCC Expansion Mode Register (GFEMR)

Table 40-8 describes GFEMR*x* fields.

Table 40-8. GFEMR*x* Field Descriptions

Bit	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode: Internal rate for PHYs[0–3] is controlled only by FTIRR[0–3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode: PHYs[0–31] are controlled by FTIRR[0–3], FIRPER, FIRSR_HI, and FIRSR_LO. Underrun status for PHYs[0–31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode mixing of internal rate and external rate is not enabled.
1	LPB	RMII Loopback diagnostic mode (Ethernet mode): 0 Normal mode 1 Loopback mode
2	CLK	RMII reference clock rate for 50 Mhz input clock from external oscillator (Ethernet mode): 0 50 MHz (for Fast Ethernet) 1 5 MHz (for 10BaseT)
3–7	—	Reserved, should be cleared.

40.18.2 FCC Ethernet Mode Register (FPSMR)

The MPC8555E supports 10/100 Mbps Ethernet through a RMII interface (according to RMII Specification March 20, 1998). The RMII use a single reference clock (50 MHz) and seven pins which are a proper subset of the MII interface pins. Ethernet features are unchanged in RMII mode. To select RMII-PHY interface, a mode bit in the Ethernet mode register (FPSMR) has been added, as shown in Figure 40-7.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	HBC	FC	SBT	LPB	LCW	FDE	MON	—		PRO	FCE	RSH	—		RMII	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1304 (FPSMR1), 0x9_1324 (FPSMR2)															

	16	20	21	22	23	24	25	26	31
Field	—		CAM	BRO	—	CRC		—	
Reset	0000_0000_0000_0000								
R/W	R/W								
Offset	0x9_1306 (FPSMR1), 0x9_1326 (FPSMR2)								

Figure 40-7. FCC Ethernet Mode Register (FPSMR*x*)

Table 40-9 describes FPSMR fields.

Table 40-9. FPSMR Ethernet Field Descriptions

Bits	Name	Description
0	HBC	Heartbeat checking 0 No heartbeat checking is performed. Do not wait for a collision after transmission. 1 Wait 40 transmit serial clocks for a collision asserted by the transceiver after transmission. TxBD[HB] is set if the heartbeat is not heard within 40 transmit serial clocks.
1	FC	Force collision 0 Normal operation 1 The channel forces a collision on transmission of every transmit frame. The MPC8555E should be configured in loopback operation when using this feature, which allows the user to test the MPC8555E collision logic. It causes the retry limit to be exceeded for each transmit frame.
2	SBT	Stop backoff timer 0 The backoff timer functions normally. 1 The backoff timer (for the random wait after a collision) is stopped whenever carrier sense is active. In this method, the retransmission is less aggressive than the maximum allowed in the IEEE 802.3 standard. The persistence (P_PER) feature in the parameter RAM can be used in combination with the SBT bit (or in place of the SBT bit).
3	LPB	Loopback operation 0 Normal operation (receiver does not receive when transmitter sends). 1 The channel is configured for internal or external loopback operation as determined by GFMR[DIAG]. For external loopback, configure DIAG for normal operation; for internal loopback configure DIAG for loopback operation.
4	LCW	Late collision window 0 A late collision is any collision that occurs at least 64 bytes from the preamble. 1 A late collision is any collision that occurs at least 56 bytes from the preamble.
5	FDE	Full-duplex Ethernet 0 Disable full duplex 1 Enable full duplex. Must be set if FSMR[LPB] is set or external loopback is performed.
6	MON	RMON mode 0 Disable RMON mode 1 Enable RMON mode

Table 40-9. FPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
7–8	—	Reserved, should be zero
9	PRO	Promiscuous 0 Check the destination address of incoming frames. 1 Receive the frame regardless of its address. A CAM can be used for address filtering when FSMR[CAM] is set.
10	FCE	Flow control enable 0 Flow control is not enabled 1 Flow control is enabled
11	RSH	Receive short frames 0 Discard short frames (frames smaller than the value specified in MINFLR). 1 Receive short frames.
12–13	—	Reserved, should be zero
14	RMII	RMII interface mode 0 MII interface 1 RMII interface. RMII to/from MII conversion logic is enabled.
15–20	—	Reserved, should be zero
21	CAM	CAM address matching 0 Normal operation. 1 Use the CAM for address matching; CAM result (16 bits) is added at the end of the frame.
22	BRO	Broadcast address 0 Receive all frames containing the broadcast address. 1 Reject all frames containing the broadcast address unless FSMR[PRO] = 1.
23	—	Reserved, should be zero
24–25	CRC	CRC selection 0x Reserved 10 32-bit CCITT-CRC (Ethernet). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1. Select this to comply with Ethernet specifications. 11 Reserved
26–31	—	Reserved, should be zero

40.18.3 Ethernet Event Register (FCCE)/Mask Register (FCCM)

The FCCE, shown in Figure 40-8, is used as the Ethernet event register when the FCC functions as an Ethernet controller. It generates interrupts and reports events recognized by the Ethernet channel. On recognition of an event, the Ethernet controller sets the corresponding FCCE bit. Interrupts generated by this register can be masked in the Ethernet mask register (FCCM).

The FCCM has the same bit format as FCCE. Setting an FCCM bit enables and clearing a bit masks the corresponding interrupt in the FCCE.

The FCCE can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values. Unmasked FCCE bits must be cleared before the CP clears the internal interrupt request.

	0–7	8	9	10	11	12	13	14	15
Field	—	GRA	RXC	TXC	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000								
R/W	R/W								
Offset	0x0x9_1310 (FCCE1), 0x0x9_1330 (FCCE2), 0x0x9_1314 (FCCM1), 0x0x9_1334 (FCCM2)								

	16–31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Offset	0x9_1312 (FCCE1), 0x9_1332 (FCCE2), 0x9_1316 (FCCM1), 0x9_1336 (FCCM2)

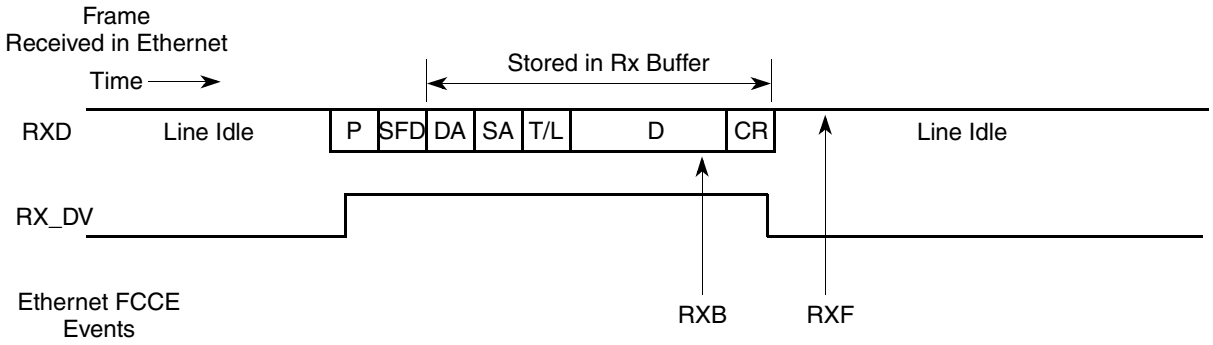
Figure 40-8. Ethernet Event Register (FCCE)/Mask Register (FCCM)

Table 40-10 describes FCCE/FCCM fields.

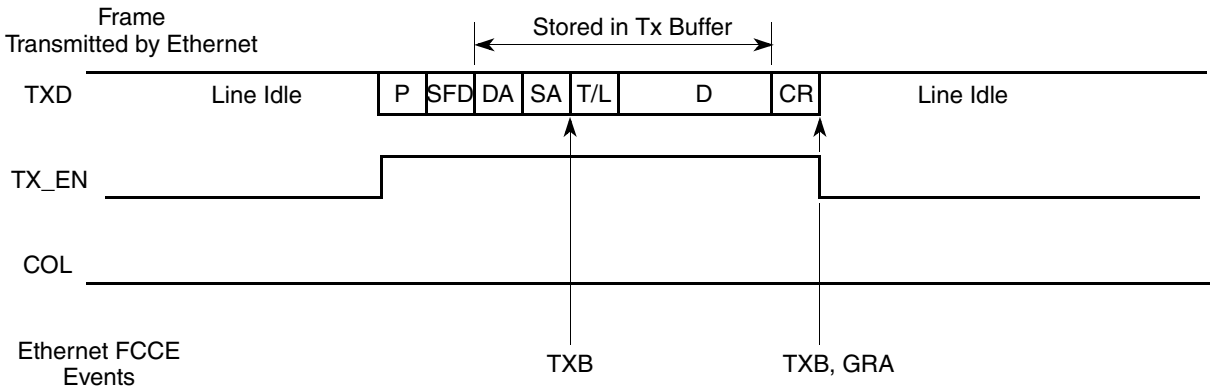
Table 40-10. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. A graceful stop, initiated by the GRACEFUL STOP TRANSMIT command, is complete. When the command is issued, GRA is set as soon the transmitter finishes sending a frame in progress. If no frame is in progress, GRA is set immediately.
9	RXC	RX control. A control frame has been received (FSMR[FCE] must be set). As soon as the transmitter finishes sending the current frame, a pause operation is performed.
10	TXC	TX control. An out-of-sequence frame was sent.
11	TXE	Tx error. An error occurred on the transmitter channel.
12	RXF	Rx frame. Set when a complete frame is received on the Ethernet channel.
13	BSY	Busy condition. Set when a frame is received and discarded due to a lack of buffers.
14	TXB	Tx buffer. Set when a buffer has been sent on the Ethernet channel.
15	RXB	Rx buffer. A buffer that was not a complete frame is received on the Ethernet channel.
16–31	—	Reserved, should be cleared.

Figure 40-9 shows interrupts that can be generated in the Ethernet protocol.



Notes:
1. RXB event assumes receive buffers are 64 bytes each.
2. The RXF interrupt may occur later than RX_DV due to receive FIFO latency.



Notes:
1. TXB events assume the frame required two transmit buffers.
2. The GRA event assumes a graceful stop transmit command was issued during frame transmission.

Legend:
P = Preamble, SFD = Start frame delimiter, DA and SA = Destination/Source address,
T/L = Type/Length, D = Data, CR = CRC bytes

Figure 40-9. Ethernet Interrupt Events Example

NOTE

The FCC status register is not valid for the Ethernet protocol. The current state of the MII signals can be read through the parallel ports.

40.19 Ethernet RxBDs

The Ethernet controller uses the RxBD to report information about the received data for each buffer. Figure 40-10 shows the FCC Ethernet RxBD format.

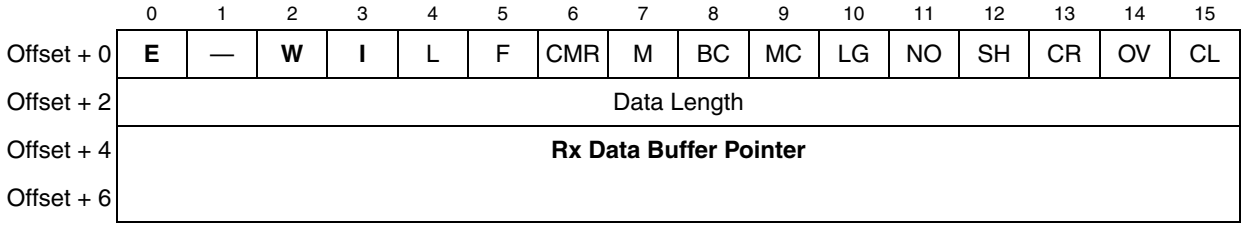


Figure 40-10. Fast Ethernet Receive Buffer (RxBD)

Table 40-11 describes Ethernet RxBD fields.

Table 40-11. RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The buffer associated with this RxBD is full or reception terminated due to an error. The core can examine or read to any fields of this RxBD. The CP does not use this BD as long as E = 0. 1 The associated buffer is empty. The RxBD and buffer are owned by the CP. Once E = 1, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (final BD in RxBD table) 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is programmable and determined only by the W bit. The RxBD table must contain more than one BD in Ethernet mode.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is used. 1 FCCE[RXB] or FCCE[RXF] are set when this buffer is used by the Ethernet controller. These two bits can cause interrupts if they are enabled.
4	L	Last in frame. Set by the Ethernet controller when this buffer is the last in a frame. This implies the end of the frame or a reception error, in which case one or more of the CL, OV, CR, SH, NO, and LG bits are set. The Ethernet controller writes the number of frame octets to the data length field. 0 Not the last buffer in a frame 1 Last buffer in a frame
5	F	First in frame. Set by the Ethernet controller when this buffer is the first in a frame. 0 Not the first buffer in a frame 1 First buffer in a frame
6	CMR	CAM match result for the frame. Set by the Ethernet controller when using a CAM for address matching and FPSMR[ECM] = 1. Valid only if the L bit is set. 0 A hit in the CAM 1 A miss in the CAM

Table 40-11. RxBD Field Descriptions (continued)

Bits	Name	Description
7	M	Miss. Set by the Ethernet controller for frames that are accepted in promiscuous mode, but are flagged as a miss by the internal address recognition. Thus, while using promiscuous mode, the user uses the miss bit to determine quickly whether the frame is destined for this station. Valid only if RxBD[I] is set. 0 The frame is received because the address is recognized. 1 The frame is received because of promiscuous mode (address is not recognized).
8	BC	Broadcast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is the broadcast address.
9	MC	Multicast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is a multicast address other than a broadcast address.
10	LG	Rx frame length violation. A frame length greater than the MFLR (maximum frame length) defined for this FCC is recognized.
11	NO	Rx nonoctet aligned frame. A frame that contained a number of bits not divisible by eight is received and the CRC check at the preceding byte boundary generated an error.
12	SH	Short frame. A frame length less than the MINFLR (minimum frame length) defined for this channel is recognized. This indication is possible only if the FPSMR[RSH] = 1.
13	CR	Rx CRC error. This frame contains a CRC error.
14	OV	Overrun. A receiver overrun occurred during frame reception.
15	CL	Collision. This frame is closed because a collision occurred during frame reception. Set only if a late collision occurs or if FPSMR[RSH] is set. The late collision definition is determined by the setting of FPSMR[LCW].

Data length is the number of octets the CP writes into this BD data buffer. It is written by the CP as the buffer is closed. When this BD is the last BD in the frame (RxBD[L] = 1), the data length contains the total number of frame octets (including four bytes for CRC). Note that at least as much memory should be allocated for each receive buffer as the size specified in MRBLR. MRBLR should be divisible by 32 and not less than 64.

The receive buffer pointer, which points to the first location of the associated data buffer, can reside in external memory. This value must be divisible by 32.

When a received frame's data length is an exact multiple of MRBLR, the last BD contains only the status and total frame length.

NOTE

At least two BDs must be prepared before beginning reception.

Figure 40-11 shows how RxBDs are used during Ethernet reception.

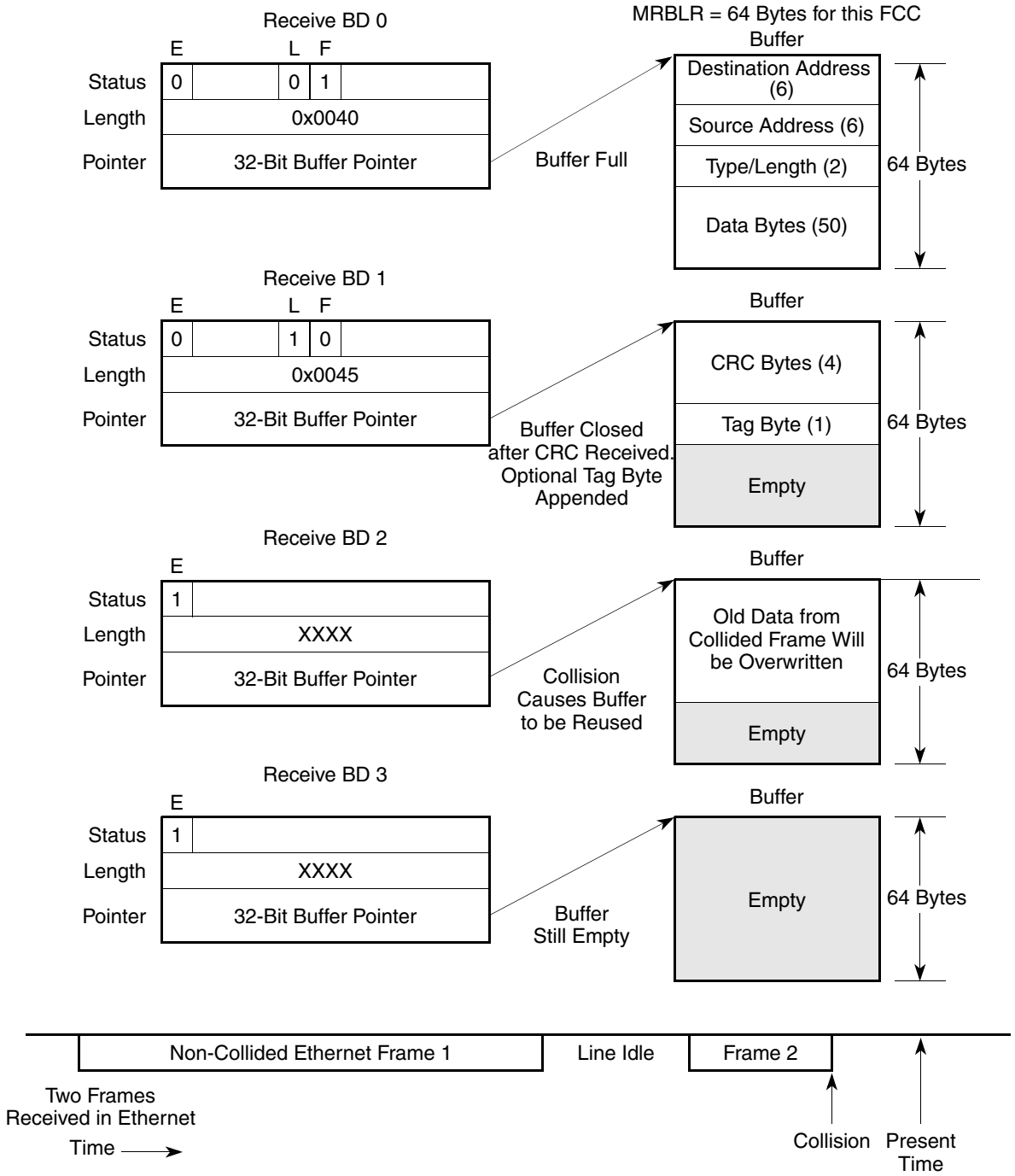


Figure 40-11. Ethernet Receiving Using RxBDs

40.20 Ethernet TxBDs

Data is sent to the Ethernet controller for transmission on an FCC channel by arranging it in buffers referenced by the channel's TxBD table. The Ethernet controller uses TxBDs to confirm transmission or

indicate errors so the core knows when buffers have been serviced. Figure 40-12 shows the FCC Ethernet TxBD format.

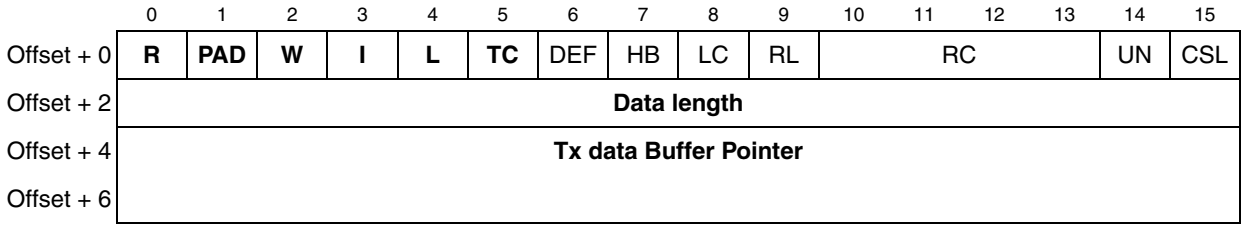


Figure 40-12. Fast Ethernet Transmit Buffer (TxBD)

Table 40-12 describes Ethernet TxBD fields.

Table 40-12. Ethernet TxBD Field Definitions

Field	Name[1]	Description
0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission; the user can manipulate this BD or its associated buffer. The CP clears R after the buffer has been sent or after an error. 1 The buffer is ready to be sent. The buffer is either waiting or in the process of being sent. The user cannot change fields in this BD or its associated buffer once R = 1.
1	**PAD**	Short frame padding. Valid only when L = 1; otherwise, it is ignored. 0 Do not add PADs to short frames. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals the MINFLR. The PAD bytes are stored in a buffer pointed to by PAD_PTR in the parameter RAM.
2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to in the table. The number of TxBDs in this table is programmable and determined only by the W bit. The TxBD table must contain more than one BD in Ethernet mode.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 FCCE[TXB] or FCCE[TXE] is set after this buffer is serviced. These bits can cause interrupts if they are enabled.
4	**L**	Last 0 Not the last buffer in the transmit frame. 1 Last buffer in the current transmit frame.
5	**TC**	Tx CRC. Valid only when the L bit is set; otherwise, it is ignored. 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. This frame did not have a collision before it was sent but it was sent late because of deferring.
7	HB	Heartbeat. The collision input is not asserted within 40 transmit serial clocks following completion of transmission. This bit cannot be set unless FPSMR[HBC] = 1. Written by the Ethernet controller after sending the associated buffer.
8	LC	Late collision. A collision occurred after the number of bytes defined in FPSMR[LCW] (56 or 64) are sent. The Ethernet controller terminates the transmission and updates LC after sending the buffer.

Table 40-12. Ethernet TxBD Field Definitions (continued)

Field	Name[1]	Description
9	RL	Retransmission limit. The transmitter failed (RET_LIM + 1) attempts to successfully send a message due to repeated collisions. The Ethernet controller updates RL after sending the buffer.
10–13	RC	Retry count. Indicates the number of retries required for this frame to be successfully sent. If RC = 0, the frame is sent correctly the first time. If RC = 15 and RET_LIM = 15 in the parameter RAM, 15 retries were needed. If RC = 15 and RET_LIM > 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The Ethernet controller updates UN after sending the buffer.
15	CSL	Carrier sense lost. Carrier sense is lost during frame transmission. The Ethernet controller updates CSL after sending the buffer.

1 **Boldfaced** entries must be initialized by the user.

Data length is the number of octets the Ethernet controller should transmit from this BD data buffer. This value should be greater than zero. The CP never modifies the data length in a TxBD.

Tx data buffer pointer, which contains the address of the associated data buffer, can be even or odd. The buffer can reside in external memory. The CP never modifies the buffer pointer.

Chapter 41
ATM Controller

The ATM controller provides the ATM and AAL layers of the ATM protocol using the universal test and operations physical layer (PHY) interface for ATM (UTOPIA level II) for both master and slave modes. It performs segmentation and reassembly (SAR) functions of AAL5, AAL2, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.

For each virtual channel (VC), the controller's ATM pace control (APC) unit generates a cell transmission rate to implement constant bit rate (CBR), variable bit rate (VBR), available bit rate (ABR), unspecified bit rate (UBR) or UBR+ traffic. To regulate VBR traffic, the APC unit performs a continuous-state leaky bucket algorithm. The APC unit also uses up to eight priority levels to prioritize real-time ATM channels, such as CBR and real-time VBR, over non-real-time ATM channels such as VBR, ABR, and UBR.

The ATM controller performs the ATM Forum (UNI-4.0) ABR flow control. To perform feedback rate adaptation, it supports forward and backward resource management (RM) cell generation and ATM Forum floating-point calculation. ABR flow control is implemented in hardware and firmware (without software intervention) to prevent potential delays during backward RM cell processing and feedback rate adaptation.

The MPC8555E supports a special mode for ATM/TDM interworking. The CPM performs automatic data forwarding without core intervention.

The MPC8555E ATM SAR controller applications are as follows:

- ATM line card controllers
- ATM-to-WAN interworking (frame relay, T1/E1 circuit emulation)
- Residential broadband network interface units (NIU) (ATM-to-Ethernet)
- High-performance ATM network interface cards (NIC)
- Bridges and routers with ATM interface

41.1 Features

The ATM controller has the following features:

- Full duplex segmentation and reassembly at 155 Mbps
- UTOPIA level II master and slave modes 8 bit
- AAL5, AAL1, AAL2, AAL0 protocols
- Up to 255 active VCs internally, and up to 64K VCs using external memory
- TM 4.0 CBR, VBR, UBR, UBR+ traffic types
- VBR type 1 and 2 traffic using leaky buckets (GCRA)
- TM 4.0 ABR flow control (EFCI and ER)

- Idle/unassign cells screening/transmission option
- External and internal rate transmit modes
- Special mode for ATM-to-port 2 or ATM-to-ATM data forwarding
- CLP and congestion indication marking
- User-defined cells up to 65 bytes
- Separate TxBD and RxBD tables for each virtual channel (VC)
- Special mode of global free buffer pools for dynamic and efficient memory allocation with early packet discard (EPD) support
- Interrupt report per channel using four priority interrupt queues
- Compliant with ATMF UNI 4.0 and ITU specification
- AAL5 cell format
  - — Reassembly
    - – Reassemble PDU directly to external memory
    - – CRC32 check
    - – CLP and congestion report
    - – CPCS_UU, CPI, and length check
    - – Abort message report
  - — Segmentation
    - – Segment PDU directly from external memory
    - – Performs PDU padding
    - – CRC32 generation
    - – Automatic last cell marking
    - – Automatic CPCS_UU, CPI, and length insertion
    - – Abort message option
- AAL1 cell format
  - — Reassembly
    - – Reassemble PDU directly to external memory
    - – Support for partially filled cells (configurable on a per-VC basis)
    - – Sequence number check
    - – Sequence number protection (CRC-3 and parity) check
  - — Segmentation
    - – Segment PDU directly from external memory
    - – Partially filled cells support (configurable on a per-VC basis)
    - – Sequence number generation
    - – Sequence number protection (CRC-3 and even parity) generation

  - Structured AAL1 cell format
    - Automatic synchronization using the structured pointer during reassembly
    - Structured pointer generation during segmentation
  - Unstructured AAL1 cell format
    - Clock recovery using external SRTS (synchronous residual time stamp) logic during reassembly
    - SRTS generation using external logic during segmentation
- AAL0 format
  - Receive
    - Whole cell is put in memory
    - CRC10 pass/fail indication
  - Transmit
    - Reads a whole cell from memory
    - CRC10 insertion option
- AAL2 format
  - Refer to Chapter 42, “ATM AAL2”
- Support for user-defined cells
  - Support cells up to 65 bytes
  - Extra header insert/load on a per-frame basis
  - Extra header size has byte resolution
  - Asymmetric cell size for send and receive
  - HEC octet insertion option
- PHY
  - UTOPIA level II supports 8 bits 25/50 MHz
    - Supports UTOPIA master and slave modes
    - Supports cell-level handshake
    - Supports multiple-PHY polling mode
- ATM pace control (APC) unit
  - Peak cell rate pacing on a per-VC basis
  - Peak-and-sustain cell rate pacing using GCRA on a per-VC basis
  - Peak-and-minimum cell rate pacing on a per-VC basis
  - Up to eight priority levels
  - Fully managed by CP with no host intervention
- Available bit rate (ABR)
  - Performs ATMF UNI 4.0 ABR flow control on a per-VC basis
  - Automatic forward-RM, backward-RM cells generation
  - Automatic feedback rate adaptation

  - Support for EFCI (explicit forward congestion indication) and ER (explicit rate)
  - RM cell floating-point calculations
  - Fully managed by CP with no host intervention
- Receive address look-up mechanism
  - Two modes of address look-up are supported
    - External CAM
    - Address compression
- OAM (operations and maintenance) cells
  - OAM filtering according to PTI field and reserved VCI field
  - Raw cell queues for transmission and reception
  - CRC-10 generation/check
  - Performance monitoring support
    - Support up to 64 bidirectional block tests simultaneously
    - Automatic FMC and BRC cell generation and termination
    - User transmit $cell_{0+1}$ count
    - User transmit $cell_0$ count
    - PM cells time stamp insertion
    - Block error detection code ($BEDC_{0+1}$) generation/check
    - Total receive $cell_{0+1}$ count
    - Total receive $cell_0$ count
  - Specifying channel code for F5 OAM cells
- ATM layer statistic gathering on a per PHY basis.
  - UTOPIA receiver error cells count (Rx parity error or short/long cells error)
  - Misinserted cell count
  - CRC-10 error cells count (ABR flow only)
- Memory management
  - RxBD table per VC with option of global free buffer pool for AAL5
  - TxBD table per VC

41.2 ATM Controller Overview

The following sections provide an overview of the transmitter and receiver portions of the ATM controller.

41.2.1 Transmitter Overview

Before the transmitter is enabled, the host must initialize the MPC8555E and create the transmit data structure, described in Section 41.10, “ATM Memory Structure.” When data is ready for transmission, the host arranges the BD table and writes the pointer of the first BD in the transmit connection table (TCT). The host issues an ATM TRANSMIT command, which inserts the current channel to the ATM pace control

(APC) unit. The APC unit controls the ATM traffic of the transmitter. It reads the traffic parameters of each channel and divides the total bandwidth among them. The APC unit can pace the peak cell rate, peak-and-sustain cell rate (GCRA traffic) or peak-and-minimum cell rate traffic. The APC implements up to eight priority levels for servicing real-time channels before non-real-time channels.

The transmitter ATM cell is 53–65 bytes and includes 4 bytes of ATM cell header, a 1-byte HEC, and 48 bytes of payload. The HEC is a constant taken from FDSR*x*[8–15]; see Section 37.5, "FCC Data Synchronization Registers (FDSRx)." User-defined cells (UDC mode) include an extra header of 1–12 bytes with an optional HEC octet. Cell transfers use the UTOPIA level II, cell-level handshake.

Transmission starts when the APC schedules a channel. According to the channel code, the ATM controller reads the channel's entry in the TCT and opens the first BD for transmission. In auto-VC-off mode, the APC automatically deactivates the current channel when no buffer is ready to transmit. In this case, a new ATM TRANSMIT command is needed for transmission of the VC to resume.

41.2.1.1 AAL5 Transmitter Overview

The transmitter reads 48 bytes from the external buffer, adds the cell header, and sends the cell through the UTOPIA interface. The transmitter adds any padding needed and appends the AAL5 trailer in the last cell of the AAL5 frame. The trailer consists of CPCS-UU+CPI, data length, and CRC-32 as defined in ITU I.363. The CPCS-UU+CPI (2-byte entry) can be specified by the user or optionally cleared by the transmitter; see Section 41.10.2.3, "Transmit Connection Table (TCT)." The transmitter identifies the last cell of the AAL5 message by setting the last (L) indication bit in the PTI field of the cell header. An interrupt may be generated to indicate the end of the frame.

When the transmission of the current frame ends and no additional valid buffers are in the BD table, the transmit process ends. The transmitter keeps polling the BD table every time this channel is scheduled to transmit. Note that a buffer-not-ready indication during frame transmission aborts the frame transfer.

41.2.1.2 AAL1 Transmitter Overview

The MPC8555E supports both structured and unstructured AAL1 formats. For the unstructured format, the transmitter reads 47 bytes from the external buffer and inserts them into the AAL1 user data field. The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is generated and inserted into the cell. The MPC8555E supports synchronous residual time stamp (SRTS) generation using external PLL. If this mode is enabled, the MPC8555E reads the SRTS code from the external logic and inserts it into four outgoing cells.

For the structured format, the transmitter reads 47 or 46 bytes from the external buffer and inserts them into the AAL1 user data field. The CP generates the AAL1 PDU header and inserts it into the cell. The header consists of the SN, SNP, and the structured pointer.

The MPC8555E supports partially filled cells configured on a per-VC basis. In this mode (TCT[PFM] = 1), only valid octets are copied from the buffer to the ATM cell; the rest of the cell is filled with padding octets.

41.2.1.3 AAL0 Transmitter Overview

No specific adaptation layer is provided for AAL0. The ATM controller reads a whole cell from an external buffer, which always contains exactly one AAL0 cell. The ATM controller optionally generates CRC10 on the cell payload and places it at the end of the payload (CRC10 field). AAL0 mode can be used to send OAM cells or AAL3/4 raw cells.

41.2.1.4 AAL2 Transmitter Overview

Refer to Section 42.3.1, "Transmitter Overview."

41.2.1.5 Transmit External Rate and Internal Rate Modes

The ATM controller supports the following two rate modes:

- External rate mode—The total transmission rate is determined by the PHY transmission rate. The FCC sends cells to keep the PHY FIFOs full; the FCC inserts idle/unassign cells to maintain the transmission rate.
- Internal rate mode—The total transmission rate is determined by the FCC internal rate timers. In this mode, the FCC does not insert idle/unassign cells. The internal rate mechanism supports up to 4 different rates. Each PHY has its own FTIRR, described in Section 41.13.5, "FCC Transmit Internal Rate Registers (FTIRRx)." The FTIRR includes the initial value of the internal rate timer. A cell transmit request is sent when an internal rate timer expires. When using internal rate mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers.

41.2.2 Receiver Overview

Before the receiver is enabled, the host must initialize the MPC8555E and create the receive data structure described in Section 41.10, "ATM Memory Structure." The host arranges a BD table for each ATM channel. Buffers for each connection can be statically allocated (that is, each BD in the BD table is associated with a fixed buffer location) or in the case of AAL5, can be fetched by the CP from a global free buffer pool. See Section 41.10.5, "ATM Controller Buffer Descriptors (BDs)."

The receiver ATM cell size is 53–65 bytes. The cell includes: 4 bytes ATM cell header, 1 byte HEC, which is ignored, and 48 bytes payload. User-defined cells (UDC mode) include an extra header of 1–12 bytes with an optional HEC octet. Cell transfers use the UTOPIA level II, cell-level handshake.

Reception starts when the PHY asserts the receive cell available signal (RxCLAV) to indicate that the PHY has a complete cell in its receive FIFO. The receiver reads a complete cell from the UTOPIA interface and translates the header address (VP/VC) to a channel code by performing an address look-up. If no matches are found, the cell is discarded and the user-network interface (UNI) statistics tables are updated. The receiver uses the channel code to read the channel parameters from the receive connection table (RCT).

41.2.2.1 AAL5 Receiver Overview

The receiver copies the 48-byte cell payload to the external buffer and calculates CRC-32 on the entire CPCS-PDU. When the last AAL5 cell arrives, the receiver checks the length, CRC-32, and CPCS-UU+CPI fields and sets the corresponding RxBD status bits. An interrupt may be generated to one

of the four interrupt queues. The receiver copies the last cell to memory including the padding and the AAL5 trailer. The CPCS-UU+CPI (16-bit entry) may be read directly from the AAL5 trailer.

The ATM controller monitors the CLP and CNG state of the incoming cells. When the message is closed, these events set RxBD[CLP] and RxBD[CNG].

When no buffer is ready to receive cells (busy state), the receiver switches to hunt state and drops all cells associated with the current frame (partial packet discard). The receiver tries to open new buffers for cell reception only after the last cell of the discarded AAL5 frame arrives.

41.2.2.2 AAL1 Receiver Overview

The ATM controller supports both AAL1 structured and unstructured formats. For the unstructured format, 47 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is checked. The MPC8555E supports SRTS clock recovery using an external PLL. In this mode, the MPC8555E tracks the SRTS from the four incoming cells and writes the SRTS code to external logic.

In the unstructured format, when the receive process begins, the receiver hunts for the first cell with a valid sequence number (SN field). When one arrives, the receiver leaves the hunt state and starts receiving. If an SN mismatch is detected, the receiver closes the RxBD, sets the sequence number error flag (RxBD[SNE]), and switches to hunt state, where it stays until a cell with a valid SN field is received.

For the structured format, 47 or 46 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of SN and SNP, is checked and the PDU status is written to the BD.

In the structured format, when the receive process begins, the receiver hunts for the first cell with a valid structured pointer to gain synchronization. When one arrives, the receiver leaves the hunt state and starts receiving. Then the receiver opens a new buffer. The structured pointer points to the first octet of the structured block, which then becomes the first byte of the new buffer. If an SN mismatch is detected, the ATM receiver closes the current RxBD, sets RxBD[SNE], and returns to the hunt state. The receiver then waits for a cell with a valid structured pointer to regain synchronization.

The MPC8555E supports partially filled cells configured on a per-VC basis. In this mode (RCT[PFM] = 1), the ATM controller copies only the valid octets from the cell user data field to the buffer.

41.2.2.3 AAL0 Receiver Overview

For AAL0, no specific adaptation layer processing is done. The ATM controller copies the whole cell to an external buffer. Each buffer contains exactly one AAL0 cell. The ATM controller calculates and checks the CRC10 of the cell payload and sets RxBD[CRE] if a CRC error occurs. AAL0 mode can be useful for receiving OAM cells or AAL3/4 raw cells.

41.2.2.4 AAL2 Receiver Overview

Refer to Section 42.4.1, “Receiver Overview.”

41.2.3 Performance Monitoring

The ATM controller supports performance monitoring testing according to ITU I.610. When performance monitoring is enabled, the ATM controller automatically generates and terminates FMCs (forward monitoring cells) and BRCs (backward reporting cells). See Section 41.6.6, "Performance Monitoring."

41.2.4 ABR Flow Control

When AAL5-ABR is enabled, the ATM controller implements the ATM Forum TM 4.0 available-bit-rate flow. It automatically inserts forward- and backward-RM cells into the user cell stream and adjusts the transmission rate according to the backwards RM cell feedback; see Section 41.10.2.2.2, "AAL5-ABR Protocol-Specific RCT." The ABR flow is controlled on a per-VC basis.

41.3 ATM Pace Control (APC) Unit

The ATM pace control (APC) unit schedules the ATM channels for transmitting. While performing this task, the APC unit uses the following parameters:

- Frequency (bandwidth) of each ATM channel
- ATM traffic pacing—Peak cell rate (PCR), sustain cell rate (SCR), and minimum rate (MCR)
- Priority level—Real-time channels (CBR or VBR-RT) are scheduled at high-priority levels; non-real-time channels (VBR-NRT, ABR, UBR) are scheduled at low-priority levels. Up to eight priority levels are available.

41.3.1 APC Modes and ATM Service Types

The ATM Forum (http://www.atmforum.com) defines the service types described in Table 41-1.

Table 41-1. ATM Service Types

Service Type	Cell Rate Pacing	Real-Time/ Non-Real-Time	Relative Priority
CBR	PCR	RT	1 (highest)
VBR-RT	PCR, SCR (peak-and-sustain)	RT	2
VBR-NRT	PCR, SCR (peak-and-sustain)	NRT	3
ABR[1]	PCR	NRT	4
UBR+	PCR, MCR (peak-and-minimum)	NRT	5
UBR	PCR	NRT	6 (lowest)

[1] When ABR flow control is active, the CP automatically adapts the APC parameters PCR, PCR_FRACTION. These parameters function as the channel's allowed cell rate (ACR).

For information about cell rate pacing, see Section 41.3.5, "ATM Traffic Type." For information about prioritization, see Section 41.3.6, "Determining the Priority of an ATM Channel."

41.3.2 APC Unit Scheduling Mechanism

The APC unit consists of an APC data structure in the dual-port RAM for each PHY and a special scheduling algorithm performed by the CP. Each PHY's APC data structure includes three elements: an APC parameter table, an APC priority table, and cell transmission scheduling tables for each priority level. (See Section 41.10.4, "APC Data Structure.")

Each PHY's APC parameter table holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The priority table holds pointers that define the location and size of each priority level's scheduling table.

Each scheduling table is divided into time slots, as shown in Figure 41-1. The user determines the number of ATM cells to be sent each time slot (cells per slot). After a channel is sent, it is removed from the current time slot and advanced to a future time slot according to the channel's assigned traffic rate (specified in time slots). The PCR parameter in the TCT, or the SCR or MCR parameters in the TCT extension (TCTE) determine the channel's actual rate.

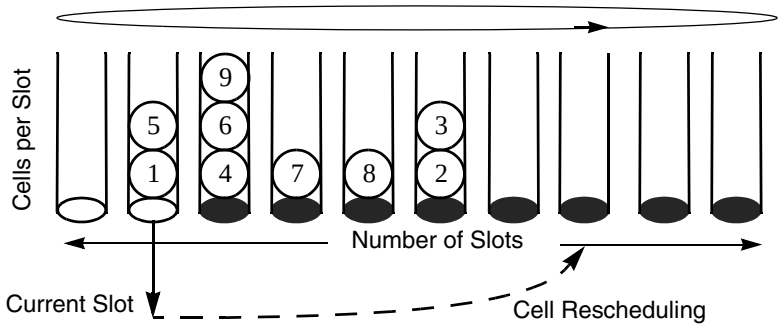


Figure 41-1. APC Scheduling Table Mechanism

Each 2-byte time-slot entry points to one ATM channel. Additional channels scheduled to transmit in the same slot are linked to each other using the APC linked-channel field in the TCT. The linked list is not limited; however, if the number of channels for the current slot exceeds the cells per slot parameter (CPS), the extra channels are sent in subsequent time slots. (The rescheduling of extra channels is based on the original slot to maintain each channel's pace.)

Note that a channel can appear only once in the scheduling table at a given time, because each channel has only one APC linked-channel field.

41.3.3 Determining the Scheduling Table Size

The following sections describe how to determine the number of cells sent per time slot and the total number of slots needed in a scheduling table.

41.3.3.1 Determining the Cells Per Slot (CPS) in a Scheduling Table

The number of cells sent per time slot is determined by the channel with the maximum bit rate; see equation A. The maximum bit rate is achieved when a channel is rescheduled to the next slot. For example, if the line rate is 155.52 Mbps and there are eight cells per slot, equation A yields a maximum VC rate of 19.44 Mbps.

(A) $$\text{Max bit rate} = \frac{\text{line rate}}{\text{cells per slot}}$$

Note that a channel can appear only once per time slot; thus, 19.44 Mbps = 155.52Mbps/8.

The cells per slot parameter (CPS) affects the cell delay variation (CDV). Because the APC unit does not put cells in a definite order within each time slot (LIFO—last-in/first-out implementation), as CPS increases, the CDV increases. However as CPS decreases, the size of the scheduling table in the dual-port RAM increases and more CPM bandwidth is required.

41.3.3.2 Determining the Number of Slots in a Scheduling Table

The number of time slots in a scheduling table is determined by the channel with the minimum bit rate; see equation B. The minimum bit rate is achieved when the channel reschedules only once in a whole table scan. (The maximum schedule advance allowed is equal to number_of_slots – 1.) For example, if the line rate is 155.52 Mbps, the minimum bit rate is 32 Kbps and the CPS is 4, then, according to equation B, the number of slots should be 1,216.

NOTE

The following APC configuration is not recommended for values outside the following ranges (PCR = peak cell rate, PCRF = peak cell rate fraction, NOS = number of slots):

PCR = 1 and PCRF = 0

PCR = NOS – 1 and PCRF = 0

(B) $$\text{Min bit rate} = \frac{\text{line rate}}{(\text{number\_of\_slots} - 1) \times \text{cells per slot}}$$

For the above example, 32 kbps = 155.52 Mbps/((1216-1) × 4).

Use equations (A) and (B) to obtain the maximum and minimum bit rates of a scheduling table. For example, given a line rate = 155.52 Mbps, number_of_slots = 1025, and CPS = 8:

$$\text{Max bit rate} = (155.52 \text{ Mbps})/8 = 19.44 \text{ Mbps}$$

$$\text{Min bit rate} = (155.52 \text{ Mbps})/(1024 \times 8) = 18.98 \text{ Kbps}$$

41.3.4 Determining the Time-Slot Scheduling Rate of a Channel

The time-slot scheduling rate of each ATM channel is defined by equation C. The resulting number of APC slots is written in either TCT[PCR], TCTE[SCR] or TCTE[MCR], depending on the traffic type.

(C) $$\text{Rate [slots]} = \frac{\text{line rate [bps]}}{\text{VC rate [bps]} \times \text{cells per slot}}$$

41.3.5 ATM Traffic Type

The APC uses the cell rate pacing parameters (PCR, SCR, and MCR) to generate CBR, VBR, ABR, UBR+, and UBR traffic. The user determines the kind of traffic that is generated per VC by writing to TCT[ATT] (ATM traffic type); see Section 41.10.2.3, "Transmit Connection Table (TCT)."

41.3.5.1 Peak Cell Rate Traffic Type

When the peak cell rate traffic type is selected, the APC schedules channels to transmit according to the PCR and PCR_FRACTION traffic parameters. Other traffic parameters do not apply to this traffic type.

41.3.5.2 Determining the PCR Traffic Type Parameters

Suppose a VC uses 15.66 Mbps of the total 155.52 Mbps and CPS = 8. Equation C yields:

$$\text{PCR [slots]} = (155.52 \text{ Mbps})/(15.66 \text{ Mbps} \times 8) = 1.241$$

The resulting number of slots is written into TCT[PCR] and TCT[PCR_FRACTION]. Because PCR_FRACTION is in units of 1/256 slots, the fraction must be converted as follows:

$$1.241 = 1+0.241 \times 256/256 = 1+ 61.79/256 \sim 1 + 62/256$$

PCR = 1 PCR_FRACTION = 62

41.3.5.3 Peak and Sustain Traffic Type (VBR)

Variable bit rate (VBR) traffic can burst at the peak cell rate as long as the long-term average rate does not exceed the sustainable cell rate. To support VBR channels, the APC implements the GCRA (generic cell rate algorithm) using three parameters—the peak cell rate (PCR), the sustained cell rate (SCR), and burst tolerance (BT), as shown in Figure 41-2. (The GCRA is also known as the leaky bucket algorithm.)

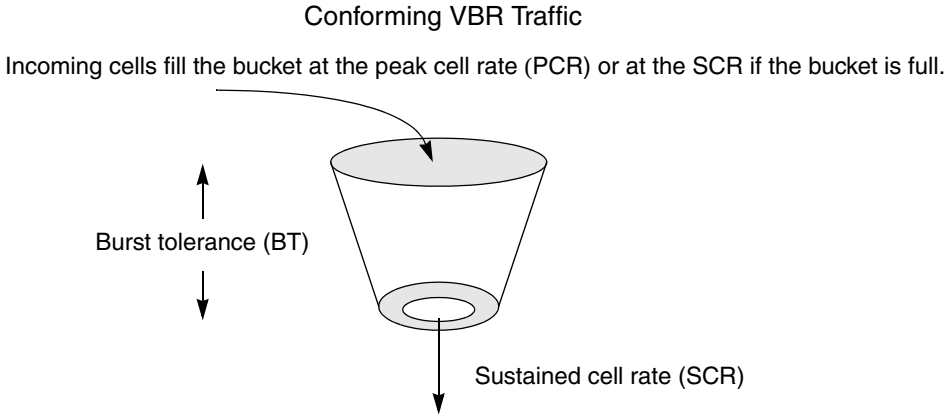


Figure 41-2. VBR Pacing Using the GCRA (Leaky Bucket Algorithm)

When a VBR channel is activated, it bursts at the peak cell rate (PCR) until reaching its initial burst tolerance (BT), which is the buffer length the network allocated for this VC. When the burst limit is reached, the APC reduces the VC's scheduling rate to the sustained cell rate (SCR). The VC continues

sending at SCR as long as TxBDs are ready. However, as each SCR time allotment elapses with no TxBD ready to send, the APC grants the VC a credit for bursting at the peak cell rate (PCR). (Gaining credit implies that the buffer at the switch is not full and can tolerate a burst transmission.) If a TxBD becomes ready, the APC schedules the VC to burst at the PCR as long as credit remains. When the burst credit ends (the network's UPC leaky bucket reaches its limit), the APC schedules the VC according to SCR.

41.3.5.3.1 Example for Using VBR Traffic Parameters

Suppose the traffic parameters of a VBR channel are PCR = 6 Mbps, SCR = 2 Mbps, MBS (maximum burst size) = 1000 cells, and CPS = 8.

Equation C (see Section 41.3.4, "Determining the Time-Slot Scheduling Rate of a Channel") yields the APC parameters, PCR, PCR_FRACTION, SCR, and SCR_FRACTION, which the user writes to the channel's TCT.

$$\text{PCR [slots]} = (155.52\ \text{Mbps})/(6\ \text{Mbps} \times 8) = 3.24$$

$$3.24 = 3 + 0.24 \times 256/256 = 3 + 61.44/256 \sim 3 + 62/256$$

$$\text{PCR} = 3 \qquad \text{PCR\_FRACTION} = 62$$

$$\text{SCR [slots]} = (155.52\ \text{Mbps})/(2\ \text{Mbps} \times 8) = 9.72$$

$$9.72 = 9 + (0.72 \times 256/256) = 9 + 184.32/256 \sim 9 + 185/256$$

$$\text{SCR} = 9 \qquad \text{SCR\_FRACTION} = 185$$

Equation D yields the number of slots the user writes to the channel's TCT[BT].

(D)
$$\begin{aligned}\text{BT [slots]} &= (\text{MBS[cells]} - 2) \times (\text{SCR[slots]} - \text{PCR[slots]}) + \text{SCR[slots]} \\ &= (1000 - 2) \times ((9+185/256) - (3+62/256)) + (9 + 185/256) \\ &= 6477\end{aligned}$$

41.3.5.3.2 Handling the Cell Loss Priority (CLP)—VBR Types 1 and 2

The MPC8555E supports two ways to schedule VBR traffic based on the cell loss priority (CLP). When TCTE[VBR2] is cleared, CLP_{0+1} cells are scheduled by PCR or SCR according to the GCRA state. When TCTE[VBR2] is set, CLP_0 cells are still scheduled by PCR or SCR according to the GCRA state, but CLP_1 cells are always scheduled by PCR. See Section 41.10.2.3.5, "VBR Protocol-Specific TCTE."

41.3.5.4 Peak and Minimum Cell Rate Traffic Type (UBR+)

To support UBR+ channels, the APC schedules transmission according to PCR and MCR. For each priority level, the APC maintains a parameter that monitors the traffic load measured as the time-slot delay between the service pointer (pointing to the current time slot waiting transmission) and a real-time slot pointer. If the transmission delay is greater than MDA (maximum delay allowed), the APC begins scheduling channels according to the MCR parameter. If the delay, however, drops below MDA, the APC again schedules channels according to the PCR. Note that in order to guarantee a minimum cell rate for UBR+ channels, there must be enough bandwidth to simultaneously send all possible channels at the MCR. See Section 41.10.2.3.6, "UBR+ Protocol-Specific TCTE."

41.3.6 Determining the Priority of an ATM Channel

The priority mechanism is implemented by adding priority table levels, which point to separate scheduling tables; see Section 41.10.4, "APC Data Structure." The APC flow control services the APC_LEVEL1 slots first. If there are no cells to send, the APC goes to the next priority level. The APC has up to eight priority levels with APC_LEVEL8 being the lowest. The user specifies the priority of an ATM channel when issuing the ATM TRANSMIT command; see Section 41.14, "ATM Transmit Command."

The real-time channels, CBR and VBR-RT, should be inserted in APC_LEVEL1; non-real-time channels, VBR-NRT, ABR, and UBR should be inserted in lower priority levels.

41.4 VCI/VPI Address Lookup Mechanism

The MPC8555E supports two ways to look up addresses for incoming cells:

- External CAM lookup
- Address compression

Writing to GMODE[ALM] (address-lookup-mechanism bit) in the parameter RAM selects the mechanism. Both mechanisms are described in the following sections.

41.4.1 External CAM Lookup

An external CAM is usually used when the range of VCI/VPI values varies widely or is unknown. Clearing GMODE[ALM] selects the external CAM address lookup mechanism. If there is no match in the external CAM, the cell is considered a misinserted cell. The external CAM can point to internal or external channels (channels whose connection table resides in external memory). The CAM input, shown in Figure 41-3, is the 32-bit cell address: PHY address, GFC + VPI, and VCI.

NOTE

The bus atomicity mechanism for CAM accesses may not function correctly when the CPM performs a DMA access to an external CAM device. This only impacts systems in which multiple CPMs will access the CAM.

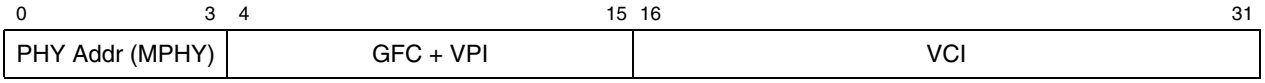


Figure 41-3. External CAM Data Input Fields

The output of the CAM, shown in Figure 41-4, is a 32-bit entry (16-bit channel code and a match-status bit).

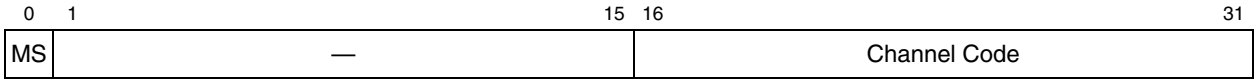


Figure 41-4. External CAM Output Fields

The external CAM fields are described in Table 41-2.

Table 41-2. External CAM Input and Output Field Descriptions

Field	Description
PHY Addr	In multiple PHY mode, this field contains the 4 least-significant bits of the current channel's physical address. Because this CAM comparison field is limited to 4 bits, two CAM devices are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the two devices. If the msb is zero, the CP accesses the CAM whose address is written in the EXT_CAM_BASE parameter in the parameter RAM; if the msb is set, the CP uses EXT_CAM1_BASE. See Section 24.4.1, "CMX UTOPIA Address Register (CMXUAR)." In single PHY mode, clear this field.
GFC+VPI, VCI	The GFC, VPI, and VCI of the current channel.
Ch Code	Pointer to internal or external connection table.
—	Reserved, should be cleared.
MS	Match status. 0 Match was found. 1 Match was not found.

41.4.2 Address Compression

The address compression mechanism uses two levels of address translation to help minimize the memory space needed to cover the available address range. The first level of translation (VP-level) uses a look-up table based on the 4-bit PHY address and the 12-bit virtual path identifier; the second level (VC-level) uses the 16-bit virtual channel identifier. If there is no match during address compression, the cell is considered a misinserted cell.

During the VP-level translation, VP_MASK in the ATM parameter RAM compresses an incoming cell's PHY address and VPI to create an index into the VP-level table. The VP-level table entry consists of another mask (VC_MASK) and a pointer to one of the VC-level tables (VCOFFSET). Note that the VP table should reside in the dual-port RAM.

In the VC-level translation, the VCI is compressed with the VC_MASK to generate a pointer to the VC-level table entry containing the received cell's channel code. The VC table should reside in external memory.

Figure 41-5 shows an example of address compression.

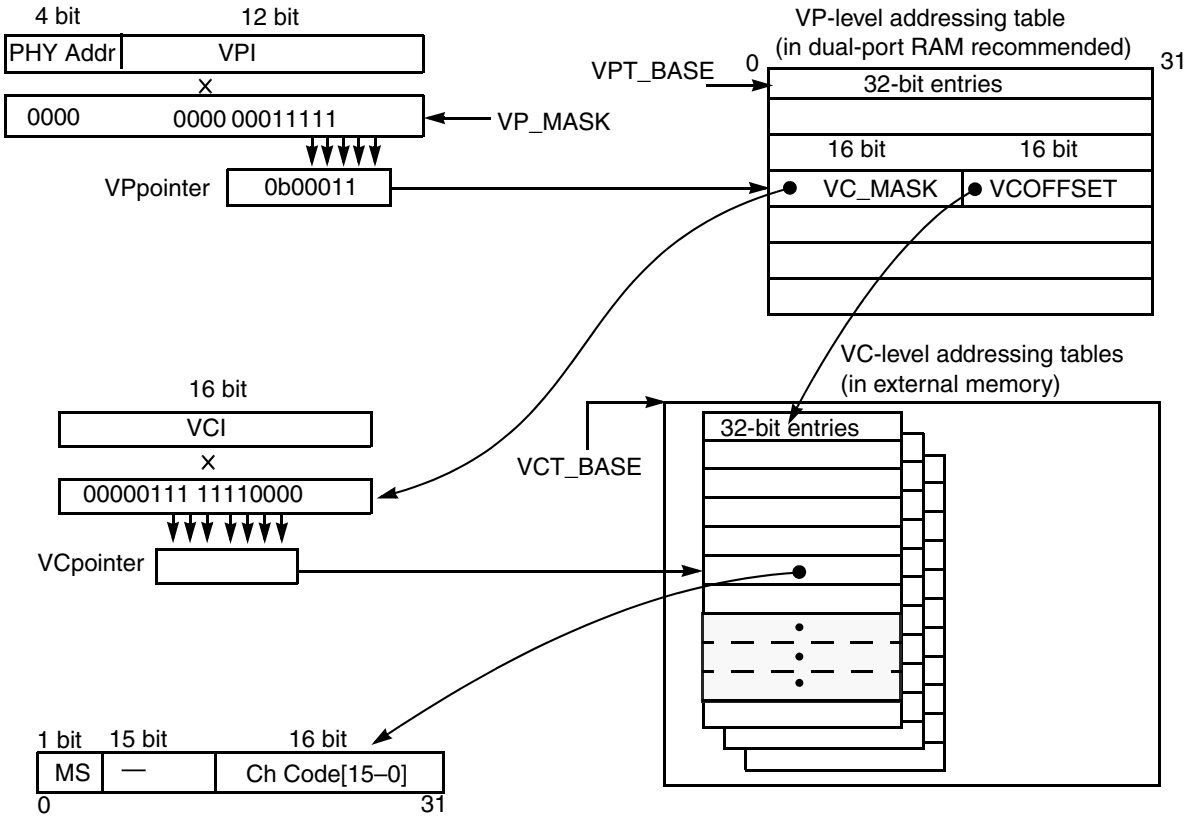


Figure 41-5. Address Compression Mechanism

Figure 41-5 shows VP_MASK selecting five VPI bits to index the VP-level table. The VP-level table entry contains the 16-bit mask (VC_MASK) and the VC-level table offset (VCOFFSET) for the next level of address mapping. The VC_MASK selects VCI bits 4–10, which is used with VCT_BASE and VCOFFSET to indicate the received cell's channel code. Address compression field descriptions are shown in Table 41-3.

Table 41-3. Field Descriptions for Address Compression

Field	Description
PHY Addr	In multiple PHY mode, this field contains the 4 least-significant bits of the current channel's physical address. Because this comparison field is limited to 4 bits, two sets of look-up tables are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the two sets of tables. If the msb is zero, the CP accesses the tables at VPT_BASE and VCT_BASE; if the msb is set, the CP uses VPT1_BASE and VCT1_BASE. See Section 24.4.1, "CMX UTOPIA Address Register (CMXUAR)." In single PHY mode, clear this field.
VCI, VPI	The VCI and VPI of the current channel.
Ch Code	Pointer to internal or external connection table.

Table 41-3. Field Descriptions for Address Compression (continued)

Field	Description
—	Reserved, should be cleared.
MS	Match status. 0 Match was found. 1 Match was not found.

41.4.2.1 VP-Level Address Compression Table (VPLT)

The size of the VP-level table depends on the number of mask bits in VP_MASK. For example, if only one PHY is available (PHY address = 0) and VPMASK = 0b11_1111_1111, VP pointer contains ten bits and the table is 4 Kbytes. Because each VPLT entry is 4 bytes, the address of an entry is VPT_BASE + VP pointer × 4.

Each VPLT entry has two parameters:

- VC_MASK—A 16-bit VC-level mask for masking the incoming cell's VCI
- VCOFFSET—A 16-bit VC-level table offset from VC_BASE that points to the appropriate VC-level table's (VCLT) starting address. The address of the VCLT is VC_BASE + VCOFFSET × 4.

  If the VCLTs are to be placed contiguously in memory, each table's VCOFFSET depends on the size of preceding tables. Each table's size depends on the number of ones in VC_MASK. Figure 41-6 gives the general formula for determining VCOFFSET.

General formula: $VCOFFSET_{(n+1)} = VCOFFSET_n + 2^{(\text{number of ones in VC\_MASKn})}$

Figure 41-6. General VCOFFSET Formula for Contiguous VCLTs

Table 41-4 shows example VCOFFSET calculations for a VP-level table with four entries.

Table 41-4. VCOFFSET Calculation Examples for Contiguous VCLTs

VP-Level Table Entry	VC_MASK	Number of Ones in VC_MASK	VC-Level Table Size	VCOFFSET
0	0x0237	6	2^6 = 64 entries	0
1	0x0230	3	2^3 = 8 entries	64
2	0xA007	5	2^5 = 32 entries	64 + 8 = 72
3	x	x	x	72 + 32 = 104

The MPC8555E can check that all unallocated bits of the PHY + VPI are 0 by setting GMODE[CUAB] (check unallocated bits) in the parameter RAM. If they are not, the cell is considered a misinserted cell.

Table 41-5 gives an example of VP-level table entry address calculation.

Table 41-5. VP-Level Table Entry Address Calculation Example

VPT_BASE	VP-Level Table Size	VP_MASK	Phy+VPI	VP Pointer	VP Entry Address
0x0024_0000	64 entries	0x0237	0x0011	0x09	VP Base = 0x240000 0x09 x 4 = 0x000024 0x240024

Figure 41-7 shows the VP pointer address compression from Table 41-5.

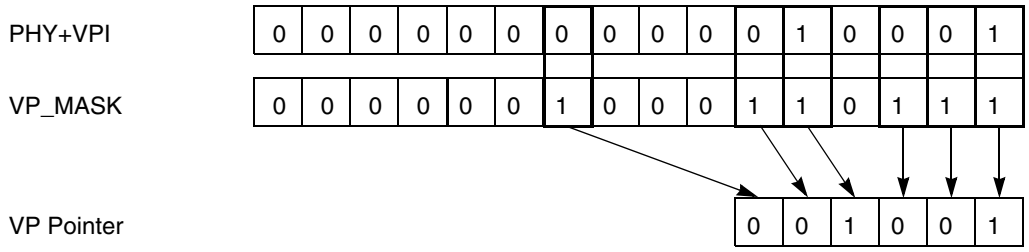


Figure 41-7. VP Pointer Address Compression

41.4.2.2 VC-Level Address Compression Tables (VCLTs)

Each VPLT entry points to a single VCLT. Like the VPLT, the size of each VCLT depends on VC_MASK. Because the VCLT contains word entries, if VC_MASK = 0b11_1111_1111, the table is 4 Kbytes. The address of an entry in this table is VCT_BASE + VCOFFSET × 4 + VCpointer × 4.

The MPC8555E can check that all unallocated VCI bits are 0 by setting GMODE[CUAB] (check unallocated bits). If they are not, the cell is considered a misinserted cell.

An example of VC-level table entry address calculation is shown in Table 41-6. Note that VCOFFSET is assumed to be 0x100 for this example.

Table 41-6. VC-Level Table Entry Address Calculation Example

VCT_BASE	VCOffset	VC-Level Table Size	VC_MASK	VCI	VC Pointer	VC Entry Address
0x0084_0000	0x0100	32 entries	0x0037	0x0031	0x19	VC Base = 0x840000 0x100 x 4 = 0x000400 0x19 x 4 = 0x000064 0x840464

Figure 41-8 shows the VC pointer address compression from Table 41-6.

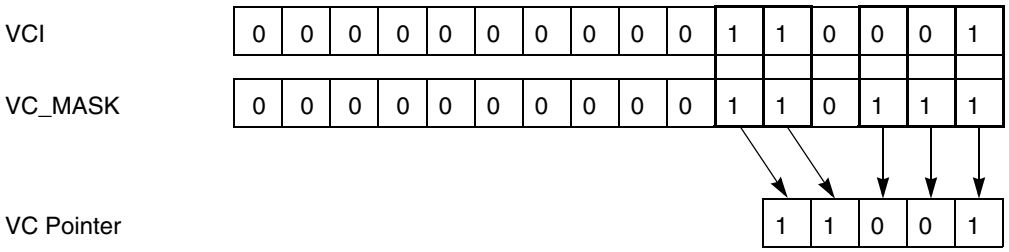


Figure 41-8. VC Pointer Address Compression

41.4.3 Misinserted Cells

If the address lookup mechanism cannot find a match (MS = 1), the cell is discarded and ATM layer statistics are updated, as described in Section 41.8, “ATM Layer Statistics.”

41.4.4 Receive Raw Cell Queue

Channel one in the RCT is reserved as a raw cell queue. The user should program channel one to operate in AAL0 protocol. The receive raw cell queue is used for removing management cells from the regular cell flow to the host. When a management cell is sent to the receive raw cell queue, the CP sets RxBD[OAM]. The ALL0 BD specifies the channel code associated with the current OAM cell.

The following are optionally removed from the regular flow and sent to the raw cell queue:

- Segment F5 OAM (PTI = 0b100). To enable F5 segment filtering, set RCT[SEGF].
- End-to-end F5 OAM (PTI = 0b101). To enable F5 end-to-end filtering, set RCT[ENDF].
- RM cells (PTI = 0b110). When ABR flow is enabled the cells are terminated internally; otherwise, they are sent to the raw cell queue.
- Reserved PTI value (PTI = 0b111). Always sent to the raw cell queue.
- VCI value: 3, 4, 6, 7–15. To enable VCI filtering set the associated bit in the VCIF entry in the parameter RAM.

Figure 41-9 shows a flowchart of the ATM cell flow.

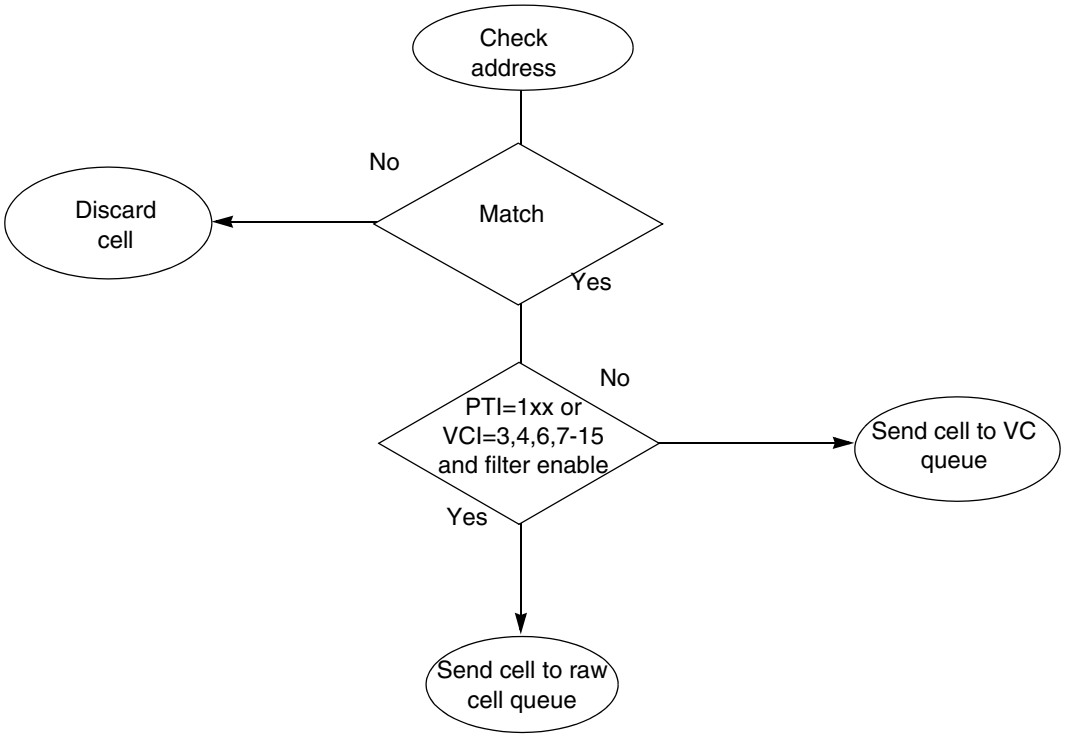


Figure 41-9. ATM Address Recognition Flowchart

NOTE

Even reserved VCI channels should appear in the CAM or address compression tables; otherwise, a cell on a reserved channel will be considered misinserted.

41.5 Available Bit Rate (ABR) Flow Control

While CBR service provides a fixed bandwidth and is useful for real-time applications with strictly bounded end-to-end cell transfer delay and cell-delay variation, ABR service is intended for data applications that can adapt to time-varying bandwidth and can tolerate significant cell transfer delay and cell delay variation. The MPC8555E implements the two following mechanisms defined by the ATM Forum TM 4.0 rate-based flow control.

- Explicit forward congestion indication (EFCI). The network supplies binary indication of whether congestion occurred along the connection path. This information is carried in the PTI field of the ATM cell header (similar to that used in frame relay). The source initially clears each ATM cell's EFCI bit, but as the cell passes through the connection, any congested node can set it. The MPC8555E detects this indication and sets the congestion indication (CI) bit in the next backwards RM cell to signal the source end station to reduce its transmission rate.
- Explicit rate (ER) feedback. The network carries explicit bandwidth information, to allow the source to adjust its rate. The source sends forward RM cells specifying its chosen transmit rate (source ER). A congested switch along the network may decrease ER to the exact rate it can

support. The destination receives forward RM cells and returns them to the source as backward RM cells. The MPC8555E implements source behavior by adjusting the rate according to each returning backward RM cell's ER.

Explicit rate feedback has several advantages over binary feedback (EFCI). Explicit rate feedback allows immediate source rate adaptation, eliminating rate oscillation caused by incremental rate changes. Using the information in RM cells, the network can allocate bandwidth evenly among active ABR channels.

41.5.1 ABR Model

Figure 41-10 shows the MPC8555E ABR model.

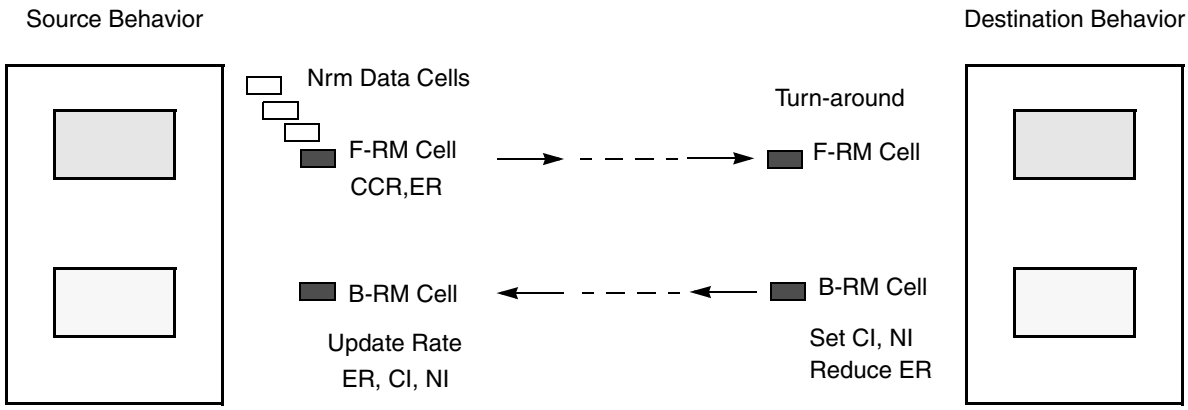


Figure 41-10. MPC8555E ABR Basic Model

The MPC8555E ABR flow control implements both source and destination behavior. The MPC8555E ABR flowchart is described in Section 41.5.1.3, "ABR Flowcharts."

41.5.1.1 ABR Flow Control Source End-System Behavior

The MPC8555E's implementation of ABR flow control for end-system sources is described in the following steps:

1. An ABR channel's allowed cell rate (ACR) lies between the minimum cell rate (MCR) and the peak cell rate (PCR).
2. ACR is initialized to the initial cell rate (ICR).
3. An F-RM (Forward-RM) cell is sent for every Nrm data cell sent. If more than Mrm cells are sent and the time elapsed since the last F-RM exceeds Trm, an F-RM cell is sent.
4. When sending an F-RM cell, the current ACR is written in the CCR (current cell rate) field of the RM cell.
5. When B-RM (backward-RM) cell is received with CI = 1 (congestion indication), ACR is reduced by ACR × RDF (rate decrease factor). After the reduction, the new ACR is determined first by letting ACRtemp be the min of (ACR, ER), and then taking the max of (ACRtemp, MCR).
6. When B-RM is received with CI=0 and NI=0 (no increase), ACR is increased by RIF × PCR (rate increase factor). The new ACR is determined first by letting ACRtemp be the min of (ACR, ER), and then taking the max of (ACRtemp, MCR).

7. Before sending an F-RM cell, if more than ADTF (ACR decrease time factor) has elapsed since sending the last F-RM cell, ACR is reduced to ICR. In other words, if the source does not fully use its gained bandwidth, it loses it and resumes sending at its initial cell rate.
8. Before sending an F-RM cell and after action 7, if more than Crm F-RM cells were sent since the last B-RM cell was received with BN=0 (backward notification), the ACR is reduced by ACR × CDF (cutoff decrease factor).
9. A source whose ACR is less than the tag cell rate (TCR) sends out-of-rate cells at the TCR. This behavior is intended for sources whose rates were set to zero by the network. These sources should periodically sense the network state by sending out-of-rate RM cells. In this case data cells will not be sent.
10. An RM cell with an incorrect CRC10 is discarded and the UNI statistics tables are updated.

41.5.1.2 ABR Flow Control Destination End-System Behavior

The MPC8555E's implementation of ABR flow control for end-system destinations is described in the following steps:

1. A received F-RM cell is turned around and sent as a B-RM cells.
2. The DIR field of the received F-RM cell is changed from 0 to 1 (backward DIR).
3. The CCR and MCR fields are taken from the F-RM and is not changed.
4. The CI bit of the B-RM cell is set if the previous data cell arrived with EFCI = 1 (congestion bit in the ATM cell header).
5. The ER field of the turn around B-RM cells is limited by TCTE[ER-BRM].
6. If a F-RM cell arrives before the previous F-RM cell was turned around (for the same connection), the new RM cell overwrites the old RM cell.

41.5.1.3 ABR Flowcharts

The MPC8555E ABR transmit and receive flow control is described in the following flowcharts. See Figure 41-11, Figure 41-12, Figure 41-13, and Figure 41-14.

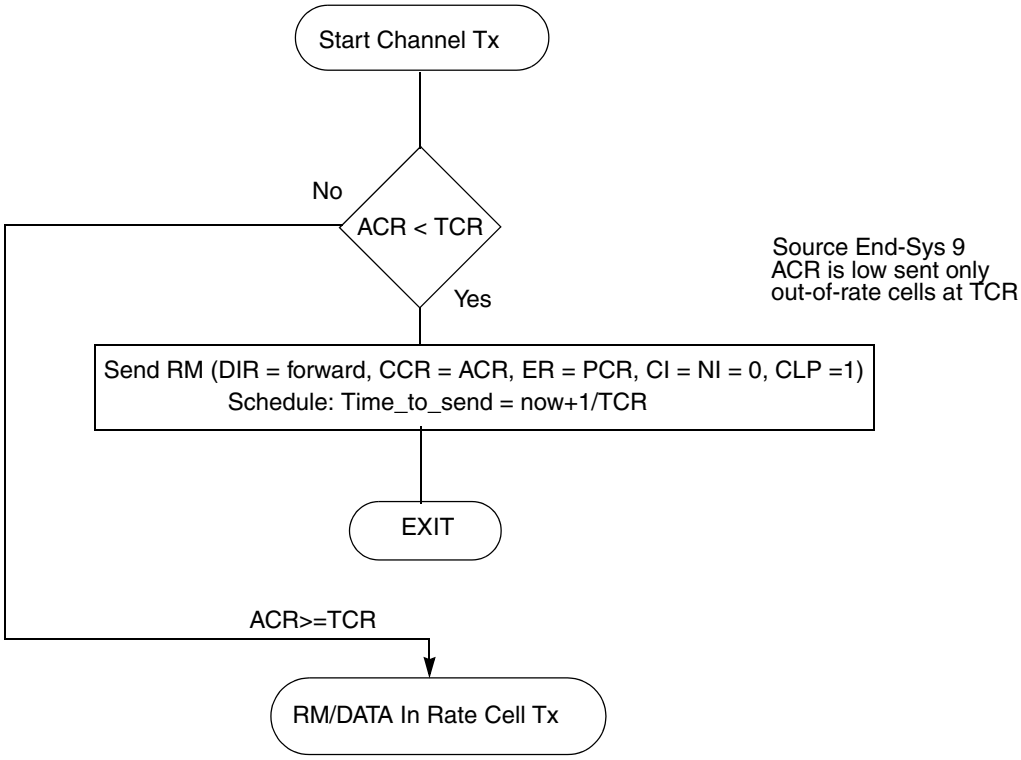


Figure 41-11. ABR Transmit Flow

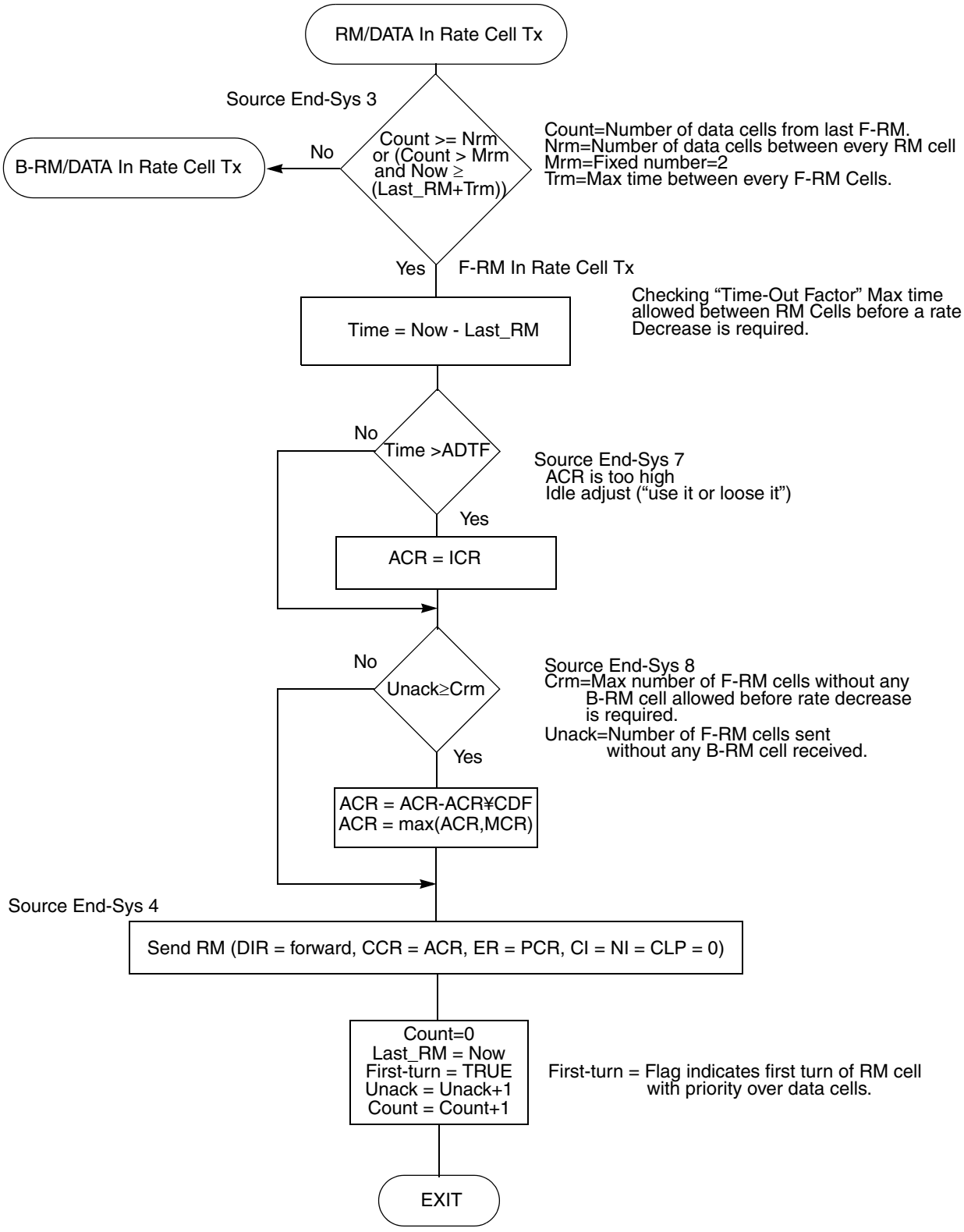


Figure 41-12. ABR Transmit Flow (continued)

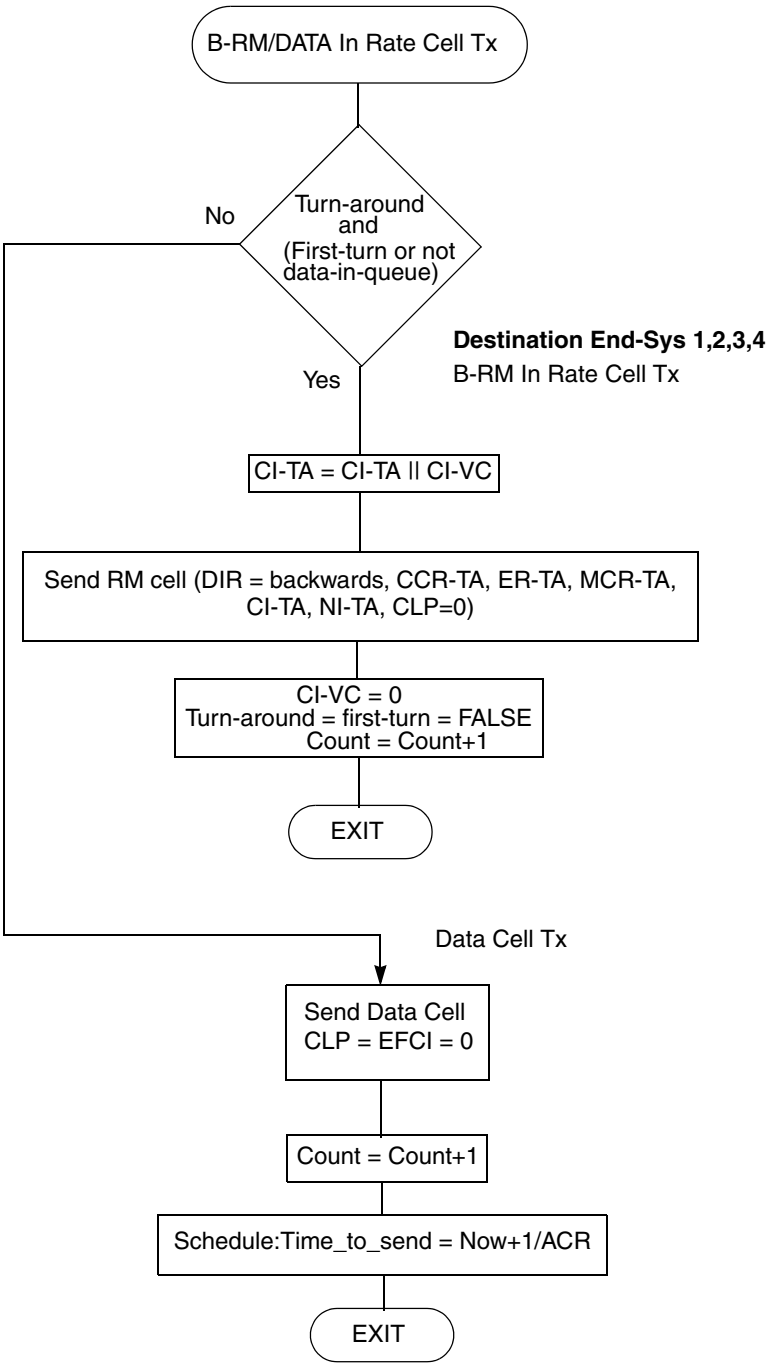


Figure 41-13. ABR Transmit Flow (continued)

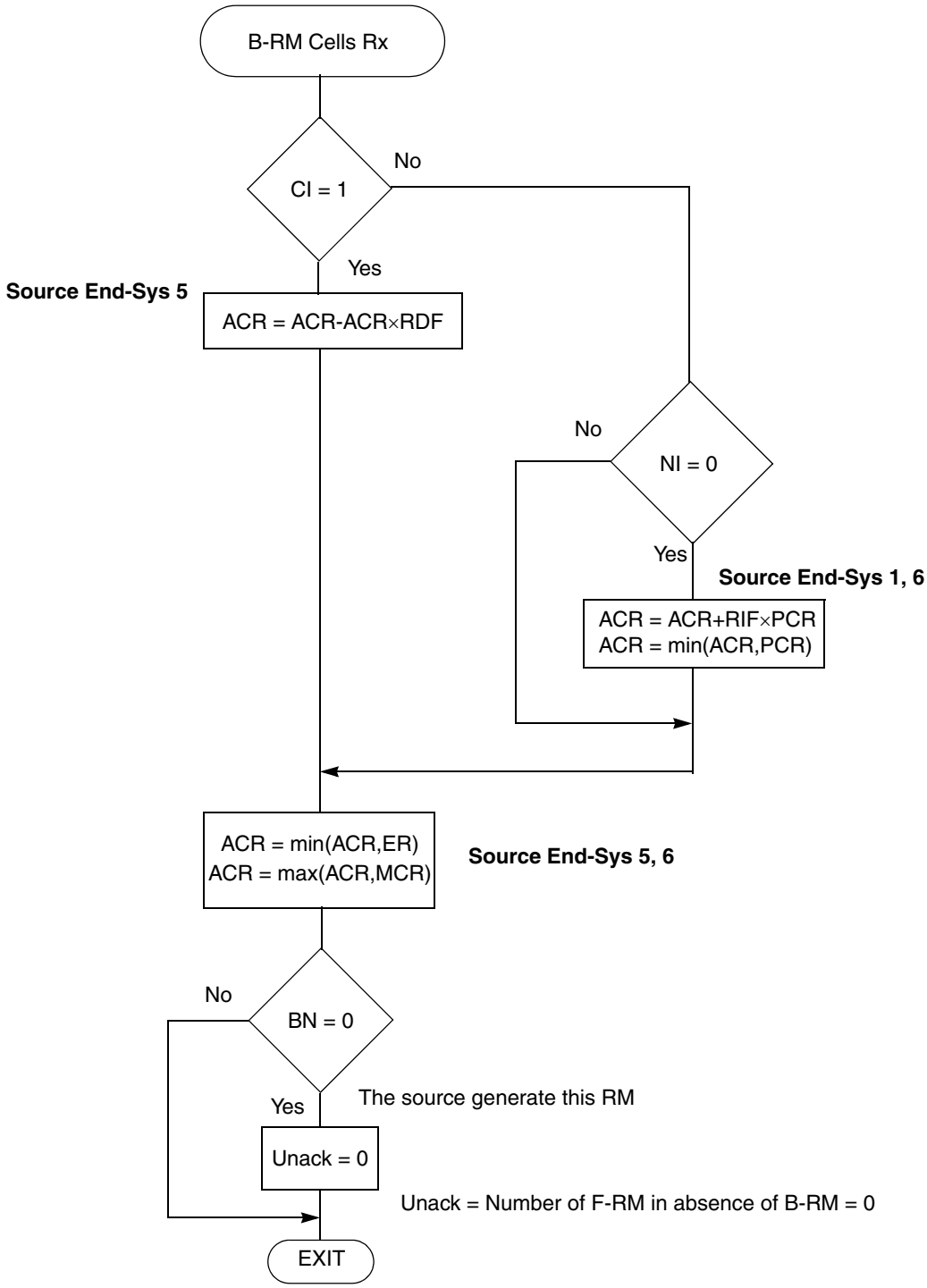


Figure 41-14. ABR Receive Flow

41.5.2 RM Cell Structure

Table 41-7 describes the structure of the RM cell supported by the MPC8555E. For more information, see the ABR flow-control traffic management specification (TM 4.0) on the ATM Forum website.

Table 41-7. Fields and Their Positions in RM Cells

Fields	Octet	Bits	Description	Value
Header	1–5	All	ATM cell header	RM-VCC PTI = 6
ID	6	All	Protocol ID	1
DIR	7	0	Direction of RM cell (0 = forward, 1 = backward)	
BN	7	1	Backward notification (BN = 0, the cell was generated by the source; BN = 1, the cell was generated by the network or by the destination)	
CI	7	2	Congestion indication. (1 = congestion, 0 = otherwise)	
NI	7	3	No increase indication. (1 = no increase allowed, 0 = otherwise)	
RA	7	4	Not used (ATM Forum ABR)	0
—	7	5–7	Reserved, should be cleared.	0
ER	8–9	All	Explicit rate; see Section 41.5.2.1	
CCR	10–11	All	Current cell rate; see Section 41.5.2.1	
MCR	12–13	All	Minimum cell rate; see Section 41.5.2.1	
QL	14–17	All	Not used (ATM Forum ABR)	0
SN	18–21	All	Not used (ATM Forum ABR)	0
—	22–51	All	Reserved, should be cleared.	0x6A for each byte
—	52	0–5	Reserved, should be cleared.	0
CRC-10	52	6–7	CRC-10	
	53	All		

41.5.2.1 RM Cell Rate Representation

Rates in the RM cells are represented in a binary floating-point format using a 5-bit exponent (e), a 9-bit mantissa (m), and a 1-bit nonzero flag (nz), as shown in Figure 41-15.

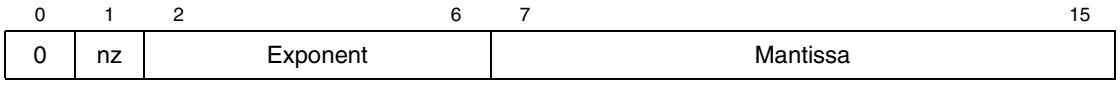


Figure 41-15. Rate Format for RM Cells

The rate (in cells/second) is calculated as in Figure 41-16.

$$\text{Rate} = \left[2^{e} \times \left(1 + \frac{m}{512}\right)\right] \times nz$$

Figure 41-16. Rate Formula for RM Cells

Initialize the traffic parameters (ER, MCR, PCR, or ICR) in the ABR protocol-specific connection tables using the rate formula in Figure 41-16.

41.5.3 ABR Flow Control Setup

Follow these steps to setup ABR flow control:

1. Initialize the ABR data structure: RCT, TCT, RCT-ABR protocol-specific, TCTE-ABR protocol-specific.
2. Initialize ABR global parameters in the parameter RAM. See Section 41.10.1, “Parameter RAM.”
3. Program the AAL-type in the RCT and TCT to AAL5 and set TCT[ABRF].

NOTE

ABR flow control is available only with AAL5.

4. The time stamp timer generates the RM cell’s time stamp, which the ABR flow control monitors to maintain source behavior in steps #3 and #7 of Section 41.5.1.1, “ABR Flow Control Source End-System Behavior.” Enable the time stamp timer by writing to the RTSCR; see Section 21.2.7, “RISC Time-Stamp Control Register (RTSCR).”
5. Initialize the ABR parameters (CPS_ABR and LINE_RATE_ABR) in the APCT; see Section 41.10.4.1, “APC Parameter Tables.” Note that when using ABR, the CPS (cells per slot) parameter in the APCPT should be a power of two.
6. Finally, send the ATM TRANSMIT command to restart channel transmission.

41.6 OAM Support

This section describes the MPC8555E’s support for ATM-layer (F4 out-of-band, and F5 in-band) operations and maintenance (OAM) of connections. Alarm surveillance, continuity checking, remote defect indication, and loopback cells are supported using OAM receive and transmit AAL0 cell queues. Using dedicated support, performance management block tests can be performed on up to 64 connections simultaneously. The CP automatically inserts forward monitoring cells (FMC) and generates backward-reporting cells (BRC) as recommended by ITU I.610.

41.6.1 ATM-Layer OAM Definitions

Table 41-8 lists pre-assigned header values at the user-network interface (UNI).

Table 41-8. Pre-Assigned Header Values at the UNI

Use	GFC	VPI	VCI	PTI	CLP
Segment OAM F4 flow cell	xxxx	aaaa_aaaa	0000_0000_0000_0011	0a0	a
End-to-end OAM F4 flow cell	xxxx	aaaa_aaaa	0000_0000_0000_0100	0a0	a
Segment OAM F5 flow cell	xxxx	aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	100	a
End-to-end OAM F5 flow cell	xxxx	aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	101	a

a = available for use by the appropriate ATM layer function

Table 41-9 lists pre-assigned header values at the network-node interface (NNI).

Table 41-9. Pre-Assigned Header Values at the NNI

Use	VPI	VCI	PTI	CLP
Segment OAM F4 flow cell	aaaa_aaaa_aaaa	0000_0000_0000_0011	0a0	a
End-to-end OAM F4 flow cell	aaaa_aaaa_aaaa	0000_0000_0000_0100	0a0	a
Segment OAM F5 flow cell	aaaa_aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	100	a
End-to-end OAM F5 flow cell	aaaa_aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	101	a

a= available for use by the appropriate ATM layer function

41.6.2 Virtual Path (F4) Flow Mechanism

The F4 flow is designated by pre-assigned virtual channel identifiers within the virtual path. The following two kinds of F4 flows can exist simultaneously:

- End-to-end (identified as VCI 4)—This flow is used for end-to-end VPC operations communications. Cells inserted into this flow can be removed only by the endpoints of the virtual path.
- Segment (identified as VCI 3)—This flow is used for communicating operations information within one VPC link or among multiple interconnected VPC links. The concatenation of VPC links is called a VPC segment. Cells inserted into this flow can be removed only by the segment endpoints, which must remove these cells to prevent confusion in adjacent segments.

41.6.3 Virtual Channel (F5) Flow Mechanism

The F5 flow is designated by pre-assigned payload type identifiers. The following two kinds of F5 flow can exist simultaneously:

- End-to-end (identified by PTI = 5)—This flow is used for end-to-end VCC operations communications. Cells inserted into this flow can be removed only by VC endpoints.
- Segment (identified by PTI = 4)—This flow is used for communicating operations information with the bound of one VCC link or multiple interconnected VCC links. A concatenation of VCC links is called a VCC segment. Segment endpoints must remove these cells to prevent confusion in adjacent segments.

41.6.4 Receiving OAM F4 or F5 Cells

OAM F4/F5 flow cells are received using the raw cell queue, described in Section 41.4.4, “Receive Raw Cell Queue.” An F4/F5 OAM cell which does not appear in the CAM or address compression tables is considered a misinserted cell.

41.6.5 Transmitting OAM F4 or F5 Cells

OAM F4/F5 flow cells are sent using the usual AAL0 transmit flow. For OAM F4/F5 cell transmission, program channel one in the TCT to operate in AAL0 mode. Enable the CR10 (CRC-10 insertion) mode as

described in Section 41.10.2.3.3, “AAL0 Protocol-Specific TCT.” Prepare the OAM F4/F5 flow cell and insert it in an AAL0 TxBD. Finally, issue a ATM TRANSMIT command to send the OAM cell. For multiple PHYs, use several AAL0 channels—each PHY should have one transmit raw cell queue that is associated with its scheduling table.

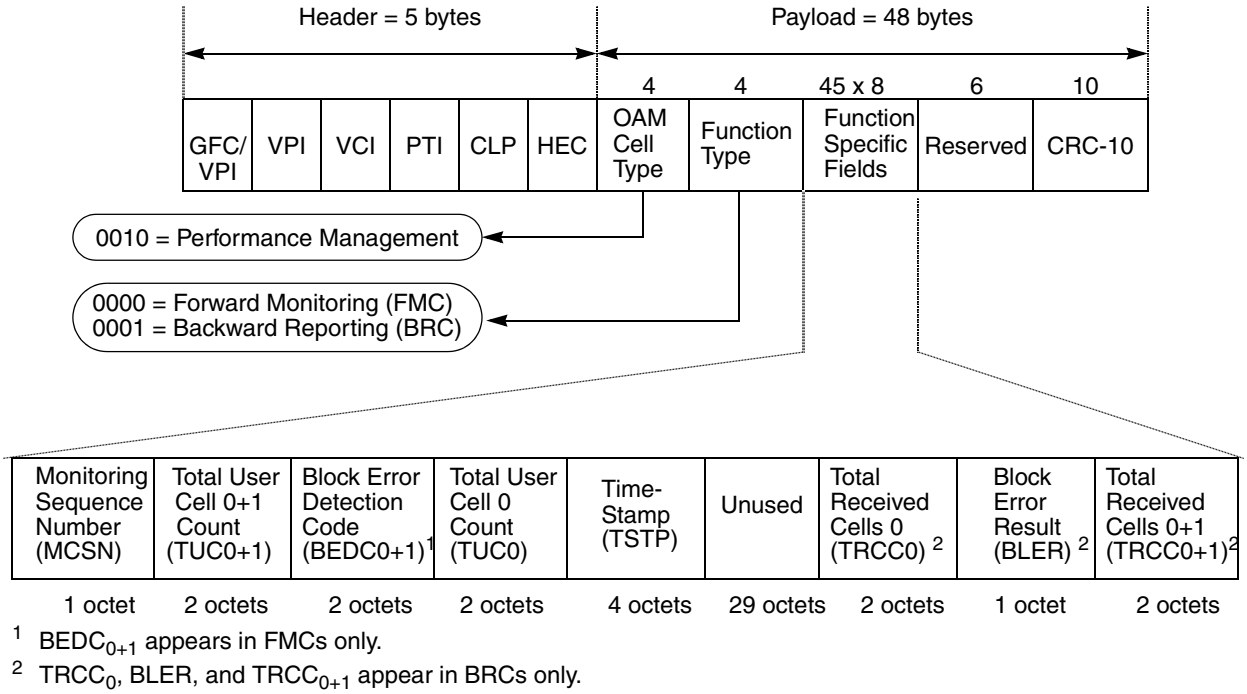
A series of OAM cells can be sent using one ATM TRANSMIT command by creating a table of AAL0 TxBDs. If the channel’s TCT[AVCF] (auto VC off) is set, the transmitter automatically removes it from the APC (that is, it does not generate periodic transmit requests for this channel after all AAL0 BDs are processed).

41.6.6 Performance Monitoring

A connection’s performance is monitored by inspecting blocks of cells (delimited by forward monitoring cells) sent between connection or segment endpoints. Each FMC contains statistics about the immediately preceding block of cells. When an endpoint receives an FMC, it adds the statistics generated locally across the same block to produce a backward reporting cell (BRC), which is then returned to the opposite endpoint.

The MPC8555E can run up to 64 bidirectional block tests simultaneously. When a bidirectional test is run, FMCs are generated for one direction and checked for the opposite.

Figure 41-17 shows the FMC and BRC cell structure.



1 $BEDC_{0+1}$ appears in FMCs only.

2 $TRCC_0$, BLER, and $TRCC_{0+1}$ appear in BRCs only.

Figure 41-17. Performance Monitoring Cell Structure (FMCs and BRCs)

Table 41-10 describes performance monitoring cell fields.

Table 41-10. Performance Monitoring Cell Fields

Field	Description	BRC	FMC
MCSN	Monitoring cell sequence number. The sequence number of the performance monitoring cell (modulo 256).	Yes	Yes
TUC_{0+1}	Total user cell 0+1 count. Counts all user cells (modulo 65,536) sent before the FMC was inserted.	Yes	Yes
TUC_0	Total user cell 0 count. Counts CLP = 0 user cells (modulo 65,536) sent before the FMC was inserted.	Yes	Yes
TSTP	Time stamp. Used to indicate when the cell was inserted.	Yes	Yes
$BEDC_{0+1}$	Block error detection code. Even parity over the payload of the block of user cells sent since the last FMC.	No	Yes
$TRCC_0$	Total received cell count 0. Counts CLP=0 user cells (modulo 65,536) received before the FMC was received.	Yes	No
BLER	Block error result. Counts error parity bits detected by the BEDC of the received FMC.	Yes	No
$TRCC_{0+1}$	Total received cell count 0+1. Counts all user cells (modulo 65,536) received before the FMC was received.	Yes	No

41.6.6.1 Running a Performance Block Test

For bidirectional PM block tests, FMCs are monitored at the receive side and generated at the transmit side. The following setup is required to run a bidirectional PM block test on an active VCC:

1. Assign one of the available 64 performance monitoring tables by writing to both RCT[PMT] and TCT[PMT] and initializing the one chosen. See Section 41.10.3, “OAM Performance Monitoring Tables.”
2. For PM F5 segment termination set RCT[SEGF]; for PM F5 end-to-end termination set RCT[ENDF].
3. Finally, set the channel’s RCT[PM] and TCT[PM] and the receive raw cell’s RCT[PM].

For unidirectional PM block tests:

- For PM block monitoring only, set only the RCT fields above.
- For PM block generation only, set only the TCT fields above.

To run a block test on a VPC, assign all the VCCs of the tested VPC to the same performance monitoring table. Configure RCT[PMT] and TCT[PMT] to specify the performance monitoring table associated with each F4 channel.

41.6.6.2 PM Block Monitoring

PM block monitoring is done by the receiver. After initialization (see Section 41.6.6.1, “Running a Performance Block Test”), whenever a cell is received for a VCC or VPC, the TRCC counters are incremented and the BEDC is calculated. When an FMC is received, the CP adds the BRC fields into the

cell payload ($TRCC_0$, $TRCC_{0+1}$, BLER) and transfers the cell to the receive raw cell queue. The user can monitor the BRC cell results and transfer the cell to the transmit raw cell queue.

Before the BRC is transferred to the transmit raw cell queue, the PM function type should be changed to backward reporting and additional checking should be done regarding the BLER field. If the sequence numbers (MCSN) of the last two FMCs are not sequential or the differences between the last two TUCs and the last two TRCCs are not equal, BLER should be set to all ones (see the ITU I.610 recommendation).

NOTE

TRCCs are free-running counters (modulo 65,536) that count user cells received. The total received cells of a particular block is the difference between TRCC values of two consecutive BRC cells. TRCC values are taken from a VC's performance monitoring table.

41.6.6.3 PM Block Generation

The transmitter generates the PM block. Each time the transmitted cell count parameter (TCC) in the performance monitoring table reaches zero, the CP inserts an FMC into the user cell stream. The CP copies the FMC header, SN-FMC, TUC_{0+1}, TUC_0, $BEDC_{0+1}$-Tx from the performance monitoring table and inserts them into the FMC payload. The TSTP value (FMC time stamp field) is taken from the MPC8555E time stamp timer; see Section 21.2.7, "RISC Time-Stamp Control Register (RTSCR)."

The TUCs are free-running counters (modulo 65,536) that count transmitted user cells. The total transmitted cells of a particular block is the difference between TUC values of two consecutive FMCs. The BEDC (BIP-16, bit interleaved parity) calculation is done on the payload of all user cells of the current tested block. The performance monitoring block can range from 1 to 2K cells, as specified in the BLCKSIZE parameter in the performance monitoring table; see Section 41.10.3, "OAM Performance Monitoring Tables."

In Figure 41-18, the performance monitoring block size is 512 cells. For every 512 user cells sent, the ATM controller automatically inserts an FMC into the regular cell stream as defined in ITU I.610. When an FMC is received, the ATM controller adds the BRC fields to the cell payload and sends the cell to the raw cell queue. The user can monitor the BRC cell results and transfer the cell to the transmit raw cell queue.

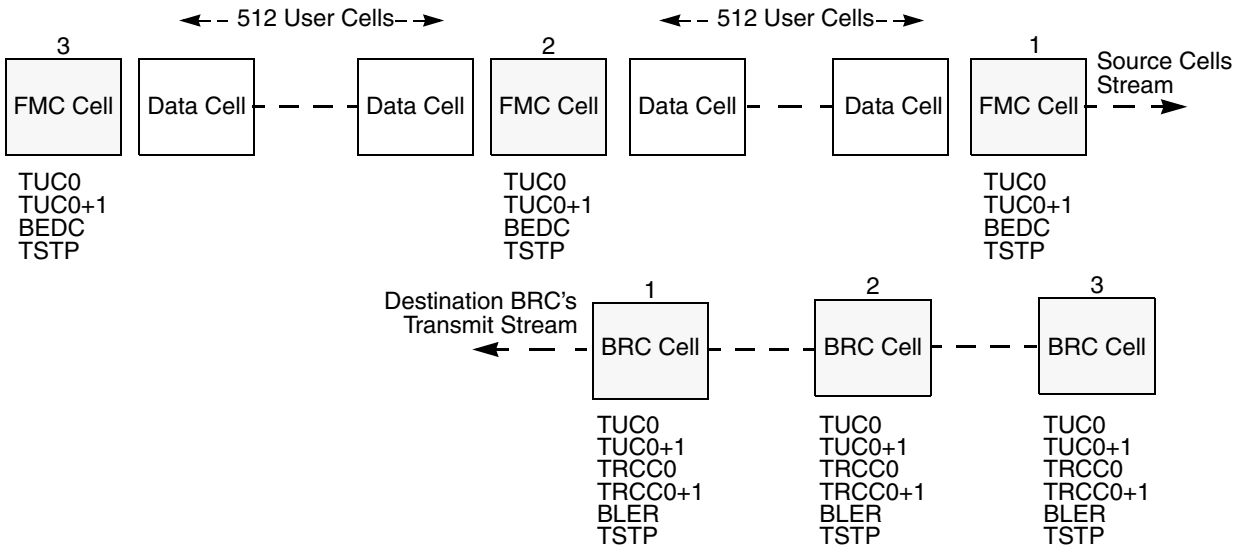


Figure 41-18. FMC, BRC Insertion

41.6.6.4 BRC Performance Calculations

BRC reception uses the regular AAL0 raw cell queue. On receiving two consecutive BRC cells, the management layer can calculate the following:

- The difference between two TUCs (Nt)
- The difference between two TRCCs (Nr)

Information about the connection can be gained by comparing Nt and Nr:

- If Nt > Nr, the difference indicates the number of lost cells of this block test.
- If Nt < Nr, the difference indicates the number of misinserted cells of this block test.
- When Nt = Nr, no cells are lost or misinserted.

41.7 User-Defined Cells (UDC)

Typical ATM cells are 53 bytes long and consist of a 4-byte header, 1-byte HEC, and 48-byte payload. The MPC8555E also supports user-defined cells with up to 12 bytes of extra header fields for internal information for switching applications. This choice is made during initialization by writing to the FPSMR; see Section 41.13.3, "FCC Protocol-Specific Mode Register (FPSMR)." As shown in Figure 41-19, the extra header size can vary between 1 to 12 bytes (byte resolution) and the HEC octet is optional.

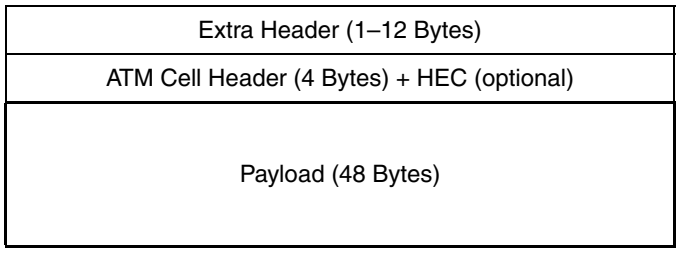


Figure 41-19. Format of User-Defined Cells

For AAL5 the extra header is taken from the Rx and TxBDs. The transmitter reads the extra header from the UDC TxBD and adds it to each ATM cell associated with the current buffer. At the receive side, the extra header of the last cell in the current buffer is written to the UDC RxBD.

For AAL0 the extra header is attached to the regular ATM cell in the buffer. The transmitter reads the extra header and the ATM cell from the buffer. The receiver writes the extra header and the regular ATM cell to the buffer.

41.7.1 UDC Extended Address Mode (UEAD)

For external CAM accesses, the UDC extra header can be used to supply extra routing information; see Figure 41-20. If GMODE[UEAD] = 1, two bytes of the UDC header are used as extensions to the ATM address and the CAM match cycle performs a double-word access. UEAD_OFFSET in the parameter RAM determines the offset from the beginning of the UDC extra header to the UEAD entry. The offset should be half-word aligned (even address). See Section 41.10.1, “Parameter RAM.”

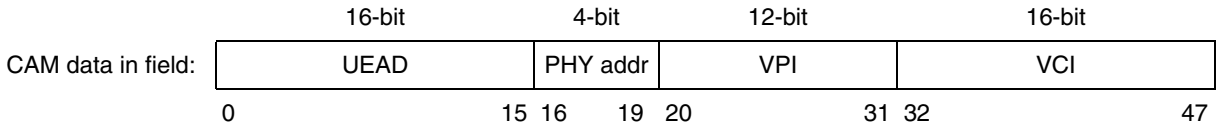


Figure 41-20. External CAM Address in UDC Extended Address Mode

41.8 ATM Layer Statistics

ATM layer statistics can be used to identify problems, such as the line-bit error rate, that affect the UNI performance. Statistics are kept in three 16-bit wrap-around counters:

- UTOPIA error dropped cells count—Counts cells discarded due to UTOPIA errors: Rx parity errors and short or long cells.
- Misinserted dropped cell count—Counts cells discarded due to address look-up failure.
- CRC10 error dropped cell count—Counts cells discarded due to CRC10 errors. (ABR only).

Counters are implemented in the dual-port RAM for each PHY device. The counters of each PHY are located in the UNI statistics table, described in Section 41.10.7, “UNI Statistics Table.”

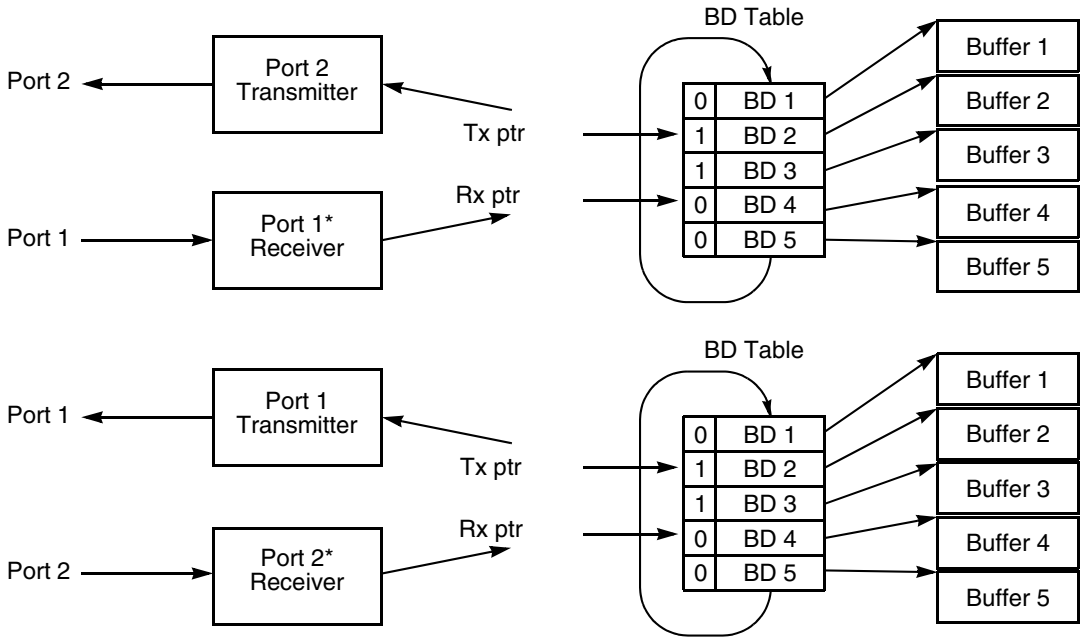
41.9 Interworking

The MPC8555E supports port-to-port interworking. Data forwarding between two ATM ports can be done in two ways:

- Core intervention—When a receive buffer in one ATM port is full and its RxBD is closed, that port interrupts the core. The core copies the port’s receive buffer pointer to the second ATM port’s TxBD and sets the ready bit (TxBD[R]). This mode is useful when additional core processing is required.
- Automatic data forwarding. This mode enables automatic data forwarding between AAL1/AAL0 and transparent mode over an ATM port. See Section 41.9.1, “ATM-to-ATM Automatic Data Forwarding.”

41.9.1 ATM-to-ATM Automatic Data Forwarding

Automatic data forwarding can be used to switch ATM AAL0 cells from one ATM port to another without core intervention. The receiver of one port and transmitter of the other port should be programed to process the same BD table. When the one port's receiver fills an AAL0 buffer, the other port's transmitter sends it, as shown in Figure 41-21.



* The receivers should be programmed to operate in opposite polarity E (empty) bit.

Figure 41-21. ATM-to-ATM Data Forwarding

As Figure 41-21 shows, when data is being forwarded from ATM port 2 to ATM port 1, the receiver of port 2 reassembles data received from a particular channel to a specific BD table. The transmitter of port 1 is programmed to operate on the same table. When port 2 fills a receive buffer, the port transmits it. The two ATM ports synchronize on port 2's RxBD[E] and port 1's TxBD[R].

When data is being forwarded from port 1 to port 2, the receiver of port 1 routes data to a specific BD table and port 2's transmitter is programmed to operate on the same table. When port 1 fills a receive buffer, the port 2 sends it. The controllers synchronize on port 1's RxBD[E] and the port 2's TxBD[R].

The receivers must be programmed to operate in opposite E-bit polarity. That is, both receivers receive data into buffers whose RxBD[E] = 0 and set RxBD[E] when a buffer is full. For the ATM receiver, set RCT[INVE] of the AAL1- and AAL0-specific areas of the receive connection table.; see Section 41.10.2.2, "Receive Connection Table (RCT)."

41.9.2 Using Interrupts in Automatic Data Forwarding

The core can program the interrupt mechanism of the ATM to trigger interrupts for events such as a buffer closing or transfer errors. The interrupt mechanism can be used to synchronize the start of the automatic bridging process. For example, to start the transmitter of a one ATM port after a specific buffer reaches the other ATM port's receiver (the buffering is required to cope with the ATM network's CDV), set RxBD[I]

in the second ATM port. When the receive buffer is full, the RxBD is closed, RxBD[E] is set (because it is operating in opposite E-bit polarity), and the core is interrupted. The core then starts the first port's transmitter.

41.10 ATM Memory Structure

The ATM memory structure, described in the following sections, includes the parameter RAM, the connection tables, OAM performance monitoring tables, the APC data structure, BD tables, and the UNI statistics table.

41.10.1 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 41-11. Note that there are additional parameters for AAL2 (refer to Table 42-13).

Table 41-11. ATM Parameter RAM Map

Offset[1]	Name	Width	Description
0x00–0x3F	—	—	Reserved, should be cleared.
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64 byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined offset from dual-port RAM base.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined offset from dual-port RAM base.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined offset from dual-port RAM base.
0x4C	—	Word	Reserved, should be cleared.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined.

Table 41-11. ATM Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. Offset to the user-defined extended address (UEAD) in the UDC extra header. Must be an even address. See Section 41.10.1.1, "Determining UEAD_OFFSET (UEAD Mode Only)." If RCT[BO] = 01, UEAD_OFFSET should be in little-endian format. For example, if the UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be programmed to 2 (second half word entry in dual-port RAM).
0x5E	—	Hword	Reserved, should be cleared.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined offset from dual-port RAM base.
0x62	APCP_BASE	Hword	APC parameter table base address. User-defined offset from dual-port RAM base.
0x64	FBT_BASE	Hword	Free buffer pool parameter table base. User-defined offset from dual-port RAM base.
0x66	INTT_BASE	Hword	Interrupt queue parameter table base. User-defined offset from dual-port RAM base.
0x68	—	—	Reserved, should be cleared.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined offset from dual-port RAM base. Note that this must be set up according to Section 29.10.7, "UNI Statistics Table>" It is not optional.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0–7] holds the 8 most-significant bits of the Rx/TxBD table base address. BD_BASE_EXT[8–31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCIF	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7-15 are received and the associated VCIF bit = 1 the cell is sent to the raw cell queue. VCIF[0–2, 5] should be zero. See Section 41.10.1.2, "VCI Filtering (VCIF)."
0x84	GMODE	Hword	Global mode. User-defined. See Section 41.10.1.3, "Global Mode Entry (GMODE)."
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 41.14, "ATM Transmit Command."
0x88		Hword	
0x8A		Hword	
0x8C	—	Word	Reserved, should be cleared.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFF_FFFF.

Table 41-11. ATM Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB_20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SNP protection look-up table base address. (AAL1 CES only.) The 32-byte table resides in dual-port RAM. AAL1_SNPT_BASE must be half-word aligned. User-defined offset from dual-port RAM base. See Section 41.10.6, "AAL1 Sequence Number (SN) Protection Table."
0x9A	—	Hword	Reserved, should be cleared.
0x9C	SRTS_BASE	Word	External SRTS logic base address. AAL1 CES only. Should be 16-byte aligned. The four least-significant bits are taken from SRTS_DEV in the AAL1-specific area of the connection table entries.
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined offset from dual-port RAM base. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP=1).
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/unassign cell size. 52 in regular mode; 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A_6A6A.
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler; see Section 21.2.7, "RISC Time-Stamp Control Register (RTSCR)." For time stamp prescaler of 1µs, program Trm to be 100 ms/1µs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less than TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate (MCR) should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be 16-byte aligned. User-defined offset from dual-port RAM base.

[1] Offset from FCC base: 0x8400 (FCC1) and 0x8500 (FCC2); see Section 21.4.2, "Parameter RAM."

41.10.1.1 Determining UEAD_OFFSET (UEAD Mode Only)

The UEAD_OFFSET value is based on the position of the user-defined extended address (UEAD) in the UDC extra header. Figure 41-22 shows how to determine UEAD_OFFSET: first determine the half-word–aligned location of the UEAD, and then read the corresponding UEAD_OFFSET value.

Offset	0 — 15	16 — 31
0x0	UEAD_OFFSET = 0x2	UEAD_OFFSET = 0x0
0x4	UEAD_OFFSET = 0x6	UEAD_OFFSET = 0x4
0x8	UEAD_OFFSET = 0xA	UEAD_OFFSET = 0x8

Figure 41-22. UEAD_OFFSETs for Extended Addresses in the UDC Extra Header

41.10.1.2 VCI Filtering (VCIF)

VCI filtering enable bits are shown in Figure 41-23.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	0	0	0	VC3	VC4	0	VC6	VC7	VC8	VC9	VC10	VC11	VC12	VC13	VC14	VC15

Figure 41-23. VCI Filtering Enable Bits

Table 41-12 describes the operation of the VCI filtering enable bits.

Table 41-12. VCI Filtering Enable Field Descriptions

Bits	Name	Description
0–2, 5	—	Clear these bits.
3, 4, 6, 7–15	VC*x*	VCI filtering enable 0 Do not send cells with this VCI to the raw cell queue. 1 Send cells with this VCI to the raw cell queue.

41.10.1.3 Global Mode Entry (GMODE)

Figure 41-24 shows the layout of the global mode entry (GMODE).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	0	0	GBL	0	0	0	ALB	CTB	REM	0	0	UEAD	CUAB	EVPT	0	ALM

Figure 41-24. Global Mode Entry (GMODE)

Table 41-13 describes GMODE fields.

Table 41-13. GMODE Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	—	Global. Asserting GBL enables snooping of connection tables. GBL should not be asserted if any of the related DMAs will access the local bus.
3–5	—	Reserved, should be cleared.
6	ALB	Address look up bus for CAM or address compression tables 0 Reside on the system bus. 1 Reside on the local bus.

Table 41-13. GMODE Field Descriptions (continued)

Bits	Name	Description
7	CTB	External connection tables bus 0 Reside on the system bus. 1 Reside on the local bus
8	REM	Receive emergency mode 0 Enable REM operation. When the receive FIFO is full, the ATM transmitter stops sending data cells until the receiver emergency state is cleared (FIFO not full). The transmitter pace is maintained, although a small CDV may be introduced. This mode enables the receiver to receive bursts of cells above the steady state performance. 1 Disable REM operation. Note that to check system performance the user may want to set this bit.
9–10	—	Reserved, should be cleared.
11	UEAD	User-defined cells extended address mode. See Section 41.7.1, “UDC Extended Address Mode (UEAD).” 0 Disable UEAD mode. 1 Enable UEAD mode.
12	CUAB	Check unallocated bits 0 Do not check unallocated bits during address compression. 1 Check unallocated bits during address compression.
13	EVPT	External address compression VP table 0 VP table resides in dual-port RAM. 1 VP table reside in external memory.
14	—	Reserved, should be cleared.
15	ALM	Address look-up mechanism. See Section 41.4, “VCI/VPI Address Lookup Mechanism.” 0 External CAM lookup. 1 Address compression.

41.10.2 Connection Tables (RCT, TCT, and TCTE)

The receive and transmit connection tables, RCT and TCT, store host-initialized connection parameters after connection set-up. These include AAL type, connection traffic parameters, BD parameters and temporary parameters used during segmentation and reassembly (SAR). The transmit connection table extension (TCTE) supports special connections that use ABR, VBR or UBR+ services. Each connection table entry resides in a 32-byte space. Table 41-14 lists sizes for RCT, TCT, and TCTE.

Table 41-14. Receive and Transmit Connection Table Sizes

ATM Service Class	RCT	TCT	TCTE
CBR, UBR service	32 bytes	32 bytes	—
ABR, VBR, UBR+ service	32 bytes	32 bytes	32 bytes

NOTE

An ATM channel is considered internal if its tables are in an internal dual-port RAM; it is considered external if its tables are in external memory.To improve performance, store parameters for fast channels in internal dual-port RAM and parameters for slower channels in external memory. Connection tables for external channels are read and written from external memory each time the CP processes a cell. The CP does, however, minimize memory access time by burst fetching the 32-byte entry and writing back only the first 24 bytes.

In all connection tables, fields which are not used must be cleared.

41.10.2.1 ATM Channel Code

Each ATM channel has a channel code used as an index to the channel's connection table entry. The first channel in the table has channel code one, the second has channel code two, and so on. Codes of 255 or less indicate internal channels; codes greater than 255 indicate external channels. Channel code one is reserved as the raw cell queue and cannot be used for another purpose. The channel code is used to specify a VC when sending a ATM TRANSMIT command, initiating the external CAM or address compression tables, and when the CP sends an interrupt to an interrupt queue.

Example:

Suppose a configuration supports 1,024 regular ATM channels. To allocate 4 Kbytes of dual-port RAM space to the internal connection table, determine that channel codes 0–63 are internal (64 VCs × 64 bytes (RCT and TCT) = 4 K). Channels 0–1 are reserved. The remaining 962 (1024 – 62) external channels are assigned channel codes 256–1217. See Figure 41-25.

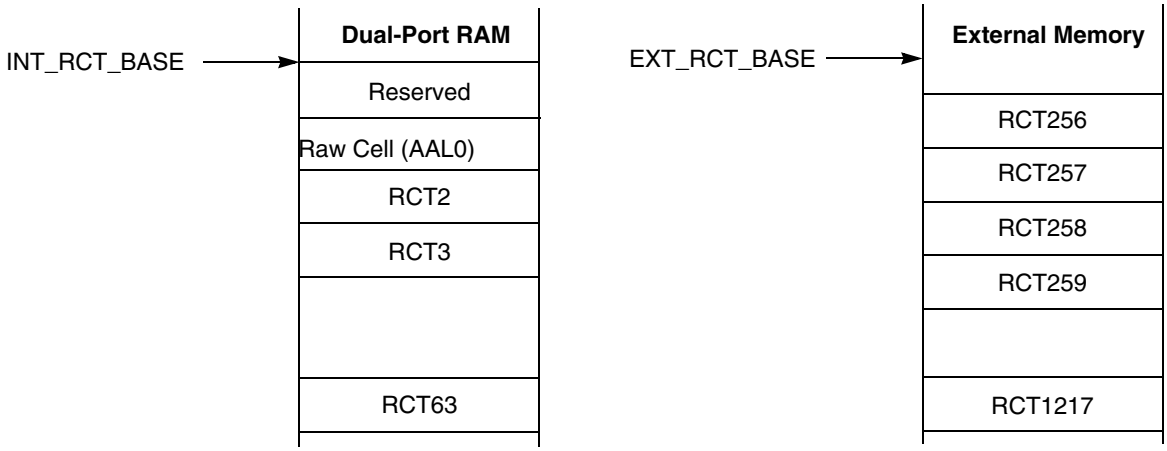


Figure 41-25. Example of a 1024-Entry Receive Connection Table

The general formula for determining the real starting address for all internal and external connection table entries is as follows:

Connection table base address + (channel code × 32)

Thus, the real starting address of the RCT entry associated with channel code 3 is as follows:

INT_RCT_BASE+ (3 × 32) = INT_RCT_BASE + 96

Even though it produces a gap in the connection table, the first external channel's real starting address of the RCT entry (channel code 256) is as follows:

EXT_RCT_BASE+ (256 × 32) = EXT_RCT_BASE + 8192

See Section 41.10.1, "Parameter RAM," to find all the connection table base address parameters. (The transmit connection table base address parameters are INT_TCT_BASE, EXT_TCT_BASE, INT_TCTE_BASE, and EXT_TCTE_BASE.)

41.10.2.2 Receive Connection Table (RCT)

Figure 41-26 shows the format of an RCT entry.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		—	DTB	BIB	—	BUFM	SEGF	ENDF	—		INTQ	
Offset + 0x02	—	INF	—										ABRF		AAL	
Offset + 0x04	RX Data Buffer Pointer (RXDBPTR)															
Offset + 0x06																
Offset + 0x08	Cell Time Stamp															
Offset + 0x0A																
Offset + 0x0C	RBD_Offset															
Offset + 0x0E	Protocol Specific See: • For AAL5,Section 41.10.2.2.1, "AAL5 Protocol-Specific RCT." • For AAL2, Section 42.4.4.1, "AAL2 Protocol-Specific RCT." • For AAL1, Section 41.10.2.2.3, "AAL1 Protocol-Specific RCT." • For AAL0, Section 41.10.2.2.4, "AAL0 Protocol-Specific RCT."															
Offset + 0x10																
Offset + 0x12																
Offset + 0x14																
Offset + 0x16																
Offset + 0x18																
Offset + 0x1A	MRBLR															
Offset + 0x1C	—		PMT						RBD_BASE							
Offset + 0x1E	RBD_BASE												—			PM

Figure 41-26. Receive Connection Table (RCT) Entry

Table 41-15 describes RCT fields.

Table 41-15. RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering—used for data buffers. 00 Reserved 01 Munged little endian 1x Big endian
	5	—	Reserved, should be cleared.
	6	DTB	Data buffers bus 0 Data buffers reside on the system bus. 1 Data buffers reside on the local bus
	7	BIB	BD, interrupt queues, free buffer pool and external SRTS logic bus 0 Reside on the system bus. 1 Reside on the local bus.
	8	—	Reserved, should be cleared.
	9	BUFM	Buffer mode. (AAL5 only) See Section 41.10.5.3, “ATM Controller Buffers.” 0 Static buffer allocation mode. Each BD is associated with a dedicated buffer. 1 Global buffer allocation mode. Free buffers are fetched from global free buffer pools.
	10	SEGF	OAM F5 segment filtering 0 Do not send cells with PTI = 100 to the raw cell queue. 1 Send cells with PTI = 100 to the raw cell queue.
	11	ENDF	OAM F5 end-to-end filtering 0 Do not send cells with PTI=101 to the raw cell queue. 1 Send cells with PTI=101 to the raw cell queue.
	12–13	—	Reserved, should be cleared.
	14–15	INTQ	Points to one of four interrupt queues available.
0x02	0	—	Internal use only. Should be cleared.
	1	INF	(AAL5 only) Indicates the receiver state. Initialize to 0 0 In idle state. 1 In AAL5 frame reception state.
	2–11	—	Internal use only. Should be cleared.
	12	ABRF	(AAL5 only). Controls ABR flow. 0 ABR flow control is disabled. 1 ABR flow control is enabled.
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol. Refer to Chapter 42, “ATM AAL2.” 101 AAL1_CES All others reserved.
0x04	—	RxDBPTR	Receive data buffer pointer. Holds real address of current position in the Rx buffer.

Table 41-15. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x08	—	Cell Time Stamp	Used for reassembly time-out. Whenever a cell is received, the MPC8555E time stamp timer is sampled and written to this field. See Section 21.2.7, "RISC Time-Stamp Control Register (RTSCR)."
0x0C	—	RBD_Offset	RxBD offset from RBD_BASE. Points to the channel's current BD. User-initialized to 0; updated by the CP.
0x0E-0x18		—	Protocol-specific area.
0x1A	—	MRBLR	Maximum receive buffer length. Used in both static and dynamic buffer allocation.
0x1C	0–1	—	Reserved, should be cleared.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	RBD_BASE	RxBD base. Points to the first BD in the channel's RxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zeros.
0x1E	0–11		
	12–14	—	Reserved, should be cleared.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC. 1 Perform performance monitoring for this VC. Whenever a cell is received for this VC the performance monitoring table that its code is written in the PMT field is updated.

41.10.2.2.1 AAL5 Protocol-Specific RCT

Figure 41-27 shows the AAL5 protocol-specific area of an RCT entry.

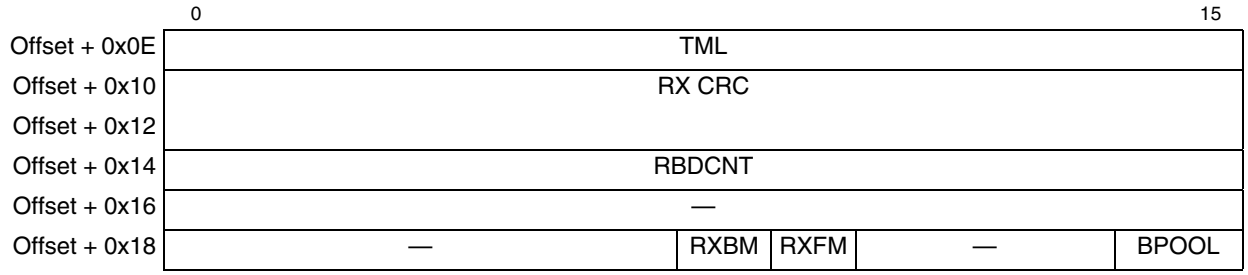


Figure 41-27. AAL5 Protocol-Specific RCT

Table 41-16 describes AAL5 protocol specific RCT fields.

Table 41-16. RCT Settings (AAL5 Protocol-Specific)

Offset	Bits	Name	Description
0x0E	—	TML	Total message length. This field is used by the CP.
0x10	—	RxCRC	CRC32 temporary result.
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. RBDCNT is initialized with MRBLR whenever the CP opens a new buffer.
0x16	—	—	Reserved, should be cleared.

Table 41-16. RCT Settings (AAL5 Protocol-Specific) (continued)

Offset	Bits	Name	Description
0x18	0–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask. Determines whether the receive buffer event is disabled. Can be changed on-the-fly. 0 The event is disabled for this channel. (The RXB event is not sent to the interrupt queue when receive buffers are closed.) 1 The event is enabled for this channel.
	9	RXFM	Receive frame interrupt mask. Determines whether the receive frame event is disabled. Can be changed on-the-fly. 0 The event is disabled for this channel. (RXF event is not sent to the interrupt queue.) 1 The event is enabled for this channel.
	10–13	—	Reserved, should be cleared.
	14–15	BPOOL	Buffer pool. Global buffer allocation mode only. Points to one of four free buffer pools. See Section 41.10.5.2.4, “Free Buffer Pool Parameter Tables.”

41.10.2.2.2 AAL5-ABR Protocol-Specific RCT

Figure 41-28 shows the AAL5-ABR protocol-specific area of an RCT entry.

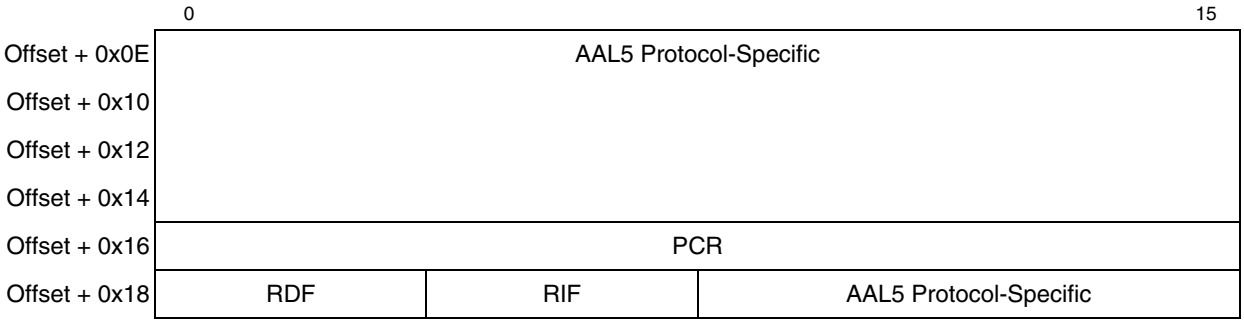


Figure 41-28. AAL5-ABR Protocol-Specific RCT

Table 41-17 describes AAL5-ABR protocol-specific RCT fields.

Table 41-17. ABR Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	—	—	AAL5 protocol-specific
0x16	—	PCR	Peak cell rate. The peak number of cells per second of the current ABR channel. The ACR (allowed cell rate) never exceeds this value. PCR uses the ATMF TM 4.0 floating-point format.
0x18	0–3	RDF	Rate decrease factor for the current ABR channel. Controls the decrease in cell transmission rate upon receipt of a backward RM cell. RDF represents a negative exponent of two, that is, the decrease factor = 2^{-RDF}. The decrease factor ranges from 1/32768 (RDF = 0xF) to 1 (RDF = 0).
	4–7	RIF	Rate increase factor of the current ABR channel. Controls the increase in the cell transmission rate upon receipt of a backward RM cell. RIF represents a negative exponent of two, that is, the increase factor = 2^{-RIF}. The increase factor ranges from 1/32768 (RIF = 0xF) to 1 (RIF = 0).
	8–15	—	AAL5 protocol-specific

41.10.2.2.3 AAL1 Protocol-Specific RCT

Figure 41-29 shows the AAL1 protocol-specific area of an RCT entry.

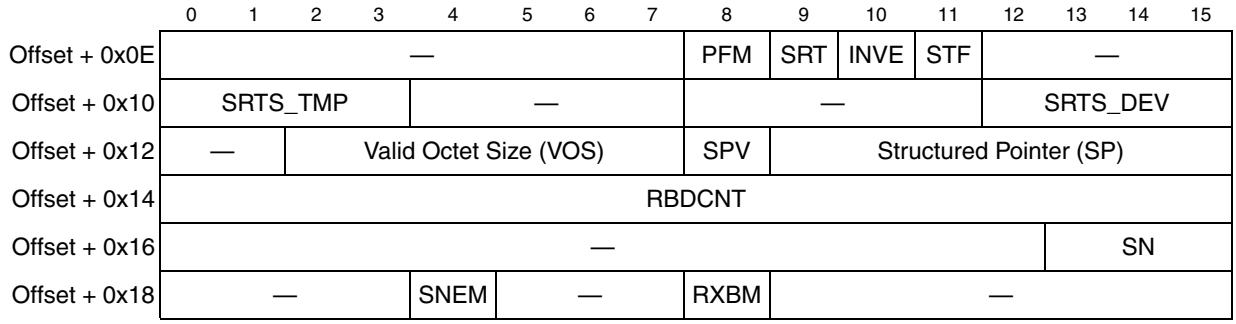


Figure 41-29. AAL1 Protocol-Specific RCT

Table 41-18 describes AAL1 protocol-specific RCT fields.

Table 41-18. AAL1 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–7	—	Reserved, should be cleared.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The receiver copies only valid octets from the AAL1 cell to the Rx buffer. The number of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8555E supports clock recovery using an external SRTS PLL. The MPC8555E tracks the SRTS from the incoming four cells with SN = 1, 3, 5, and 7 and writes it to the external SRTS device. Every eight cells the CP writes a valid SRTS to external logic. 0 SRTS mode is not used. 1 SRTS mode is used.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12–15	—	Reserved, should be cleared.
0x10	0–3	SRTS_TMP	Used by the CP to store the received SRTS code. After a cell with SN = 7 is received, the CP writes the SRTS code to the external SRTS device.
	4–11	—	Reserved, should be cleared.
	12–15	SRTS_DEV	Selects an SRTS device, whose address is SRTS_BASE[0–27] + SRTS_DEV[28–31]. The 16 byte-aligned SRTS_BASE is taken from the parameter RAM.

Table 41-18. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x12	0–1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured, service values 1–47 are valid; for structured service, values 1-46 are valid. Partially filled cell mode only.
	8	SPV	Structured pointer valid. Should be user-initialized user to zero. Structured format only.
	9–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. This field should be initialized by the user to zero. Used in structured format only.
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. Initialized with MRBLR whenever the CP opens a new buffer.
0x16	0–12	—	Reserved, should be cleared.
	13–15	SN	Sequence number. Used by the CP to check incoming cell's sequence number.
0x18	0–3	—	Reserved, should be cleared.
	4	SNEM	Sequence number error flag interrupt mask 0 This mode is disabled. 1 When an out-of-sequence error occurs, an RXB interrupt is sent to the interrupt queue even if RCT[RXBM] is cleared. Note that this mode is the buffer error reporting mechanism during automatic data forwarding (ATM-to-TDM bridging) when no buffer processing is required (RCT[RXBM]=0).
	5–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is disabled. (The event is not sent to the interrupt queue.) 1 The receive buffer event of this channel is enabled.
	9–15	—	Reserved, should be cleared.

41.10.2.2.4 AAL0 Protocol-Specific RCT

Figure 41-30 shows the layout for the AAL0 protocol-specific RCT.

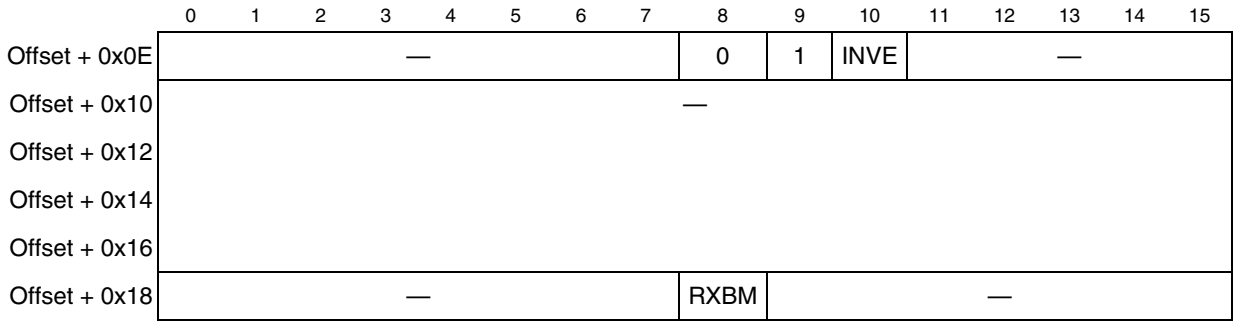


Figure 41-30. AAL0 Protocol-Specific RCT

Table 41-19 describes AAL0 protocol specific RCT fields.

Table 41-19. AAL0-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0-7	—	Reserved, should be cleared.
	8-9	0b01	Must be programmed to 0b01 for AAL0.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11-15	—	Reserved, should be cleared.
0x10	—	—	Reserved, should be cleared.
0x18	0–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is masked. (The RXB event is not sent to the interrupt queue when receive buffers are closed.) 1 The receive buffer event of this channel is enabled.
	9–15	—	Reserved, should be cleared.

41.10.2.2.5 AAL2 Protocol-Specific RCT

Refer to Section 42.4.4.1, “AAL2 Protocol-Specific RCT.”

41.10.2.3 Transmit Connection Table (TCT)

Figure 41-31 shows the format of an TCT entry.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		—	DTB	BIB	AVCF	—	ATT		CPUU	VCON	INTQ	
Offset + 0x02	—	INF	—										ABRF	AAL		
Offset + 0x04	Tx Data Buffer Pointer (TXDBPTR)															
Offset + 0x06																
Offset + 0x08	TBDCNT															
Offset + 0x0A	TBD_OFFSET															
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															
Offset + 0x10	Protocol Specific • For AAL5, see Section 41.10.2.3.1, “AAL5 Protocol-Specific TCT.” • For AAL2, see Section 42.3.5.1, “AAL2 Protocol-Specific TCT.” • For AAL1, see Section 41.10.2.3.2, “AAL1 Protocol-Specific TCT.” • For AAL0, see Section 41.10.2.3.3, “AAL0 Protocol-Specific TCT.”															
Offset + 0x12																
Offset + 0x14																
Offset + 0x16	APC Linked Channel (APCLC)															
Offset + 0x18	ATM Cell Header (VPI,VCI,PTI,CLP)															
Offset + 0x1a																
Offset + 0x1C	—		PMT						TBD_BASE							
Offset + 0x1E	TBD_BASE												BNM	STPT	IMK	PM

Figure 41-31. Transmit Connection Table (TCT) Entry

Table 41-20 describes general TCT fields.

Table 41-20. TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering. This field is used for data buffers. 00 Reserved 01 Power PC little endian 1x Big endian
	5	—	Reserved, should be cleared.
	6	DTB	Data buffer bus 0 Reside on the system bus. 1 Reside on the local bus.

Table 41-20. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
	7	BIB	BD, interrupt queue and external SRTS logic bus 0 Reside on the system bus. 1 Reside on the local bus. **Note:** When using AAL5, AAL1 CES in UDC mode, BDs and data should be placed on the same bus (TCT[DTB] = TCT[BIB]).
	8	AVCF	Auto VC off. Determines the behavior of the APC when the last buffer associated with this VC has been sent and no more ready buffers are in the VC's TxBD table. 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears TCT[VCON]. **Note:** When over-subscribing UBR or UBR+ channels, set AVCF so that the CPM does not become overloaded polling non-active VCs.
	9	—	Reserved, should be cleared.
	10–11	ATT	ATM traffic type 00 Peak cell-rate pacing. The host must initialize PCR and the PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and minimum cell rate pacing (UBR+ traffic). The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved
	12	CPUU	CPCS-UU+CPI insertion (used for AAL5 only). 0 CPCS-UU+CPI insertion disabled. The transmitter clears the CPCS-UU+CPI fields. 1 CPCS-UU+CPI insertion enabled. The transmitter reads the CPCS-UU+CPI (16-bit entry) from external memory. It should be placed after the end of the last buffer (it should not be included in the buffer length).
	13	VCON	Virtual channel is on. Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPT] (stop transmit), the CP deactivates this channel and clears VCON when the channel is next encountered in the APC scheduling table. The host can issue another ATM TRANSMIT command only after the CP clears VCON.
	14–15	INTQ	Points to one of four interrupt queues available.
0x02	0	—	Internal use only. Should be cleared.
	1	INF	Used for AAL5 Only. Indicates the transmitter state. Initialize to 0 0 In idle state. 1 In AAL5 frame transmission state.
	2–11	—	Internal use only. Should be cleared.
	12	ABRF	Used for AAL5 Only. 0 ABR Flow control is disabled. 1 ABR Flow control is enabled.

Table 41-20. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol. Refer to Chapter 42, "ATM AAL2." 101 AAL1_CES All others reserved.
0x04	—	TxDBPTR	Tx data buffer pointer. Holds the real address of the current position in the Tx buffer.
0x08	—	TBDCNT	Transmit BD count. Counts the remaining data to transmit in the current transmit buffer. Its initial value is loaded from the data length field of the TxBD when a new buffer is open; its value is subtracted for any transmitted cell associated with this channel.
0x0A	—	TBD_OffSet	Transmit BD offset. Holds offset from TBD_BASE of the current BD. Should be cleared initially.
0x0C	0–7	Rate Reminder	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared initially.
	8–15	PCR Fraction	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	—	Protocol-specific
0x16	—	APCLC	APC linked channel. Used by the CP. Initialize to 0 (null pointer).
0x18	—	ATMCH	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.
0x1C	0–1	—	Reserved, should be cleared.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	TBD_BASE	TxBD base. Points to the first BD in the channel's TxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zero.
0x1E	0–11		
	12	BNM	Buffer-not-ready interrupt mask. Can be changed on-the-fly. 0 The transmit buffer-not-ready event of this channel is masked. (TBNR event is not sent to the interrupt queue.) 1 The buffer-not-ready event of this channel is enabled.
	13	STPT	Stop transmit. Should be cleared initially. When the host sets this bit, the CP deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. Note that for AAL5 if STPT is set and frame transmission is already started (TCT[INF]=1), an abort indication will be sent (last cell with zero length field).
0x1E	14	IMK	Interrupt mask. Can be changed on-the-fly. 0 The transmit buffer event of this channel is masked. (TXB event is not sent to the interrupt queue.) 1 The transmit buffer event of this channel is enabled.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC. 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

41.10.2.3.1 AAL5 Protocol-Specific TCT

Figure 41-32 shows the AAL5 protocol-specific TCT.

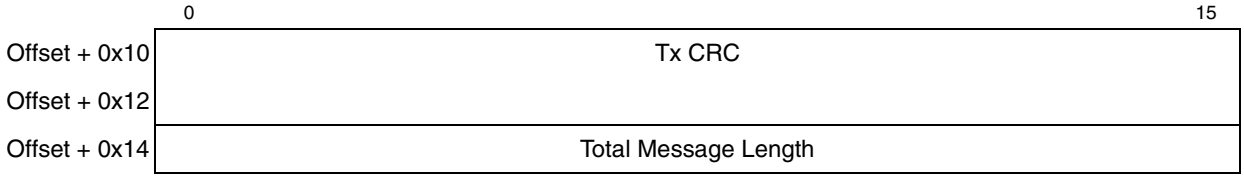


Figure 41-32. AAL5 Protocol-Specific TCT

Table 41-21 describes AAL5 protocol-specific TCT fields.

Table 41-21. AAL5-Specific TCT Field Descriptions

Offset	Name	Description
0x10	Tx CRC	CRC32 temporary result.
0x14	Total Message Length	This field is used by the CP.

41.10.2.3.2 AAL1 Protocol-Specific TCT

Figure 41-33 shows the AAL1 protocol-specific TCT.

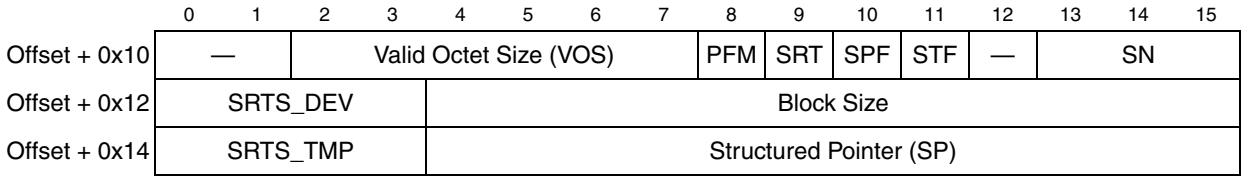


Figure 41-33. AAL1 Protocol-Specific TCT

Table 41-22 describes AAL1 protocol-specific TCT fields.

Table 41-22. AAL1 Protocol-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0-1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Partially filled cell mode only. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured service, values 1–47 are valid; for structured service, values 1–46 are valid.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The transmitter copies only valid octets from the buffer to the AAL1 cell. The size of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8555E supports SRTS generation using external logic. If this mode is enabled, the MPC8555E reads the SRTS from external logic and inserts it into four cells for which SN = 1, 3, 5, or 7. The MPC8555E reads the new SRTS from external logic every eight cells. 0 SRTS mode is not used. 1 SRTS mode is used.
	10	SPF	Structured pointer flag. Indicates that a structured pointer has been inserted in the current block. The user should initialize this field to zero. Used by the CP only.
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12	—	Reserved, should be cleared.
	13–15	SN	Sequence number field. Used by the CP to check the incoming cells SN. Should be cleared initially.
0x12	0–3	SRTS_DEV	Used to select a SRTS device. The SRTS device address is SRTS_BASE[0–27]+SRTS_DEV[28:31]. SRTS_BASE is taken from the parameter RAM and is 16-byte aligned.
	4–15	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum).
0x14	0–3	SRTS_TMP	Before a cell with SN = 1 is sent, the CP reads the SRTS code from external SRTS logic, writes it to SRTS_TMP, and then inserts SRTS_TMP into the next four cells with an odd SN.
	4–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. Should be cleared initially. Structured format only.

41.10.2.3.3 AAL0 Protocol-Specific TCT

Figure 41-34 shows the AAL0 protocol-specific TCT.

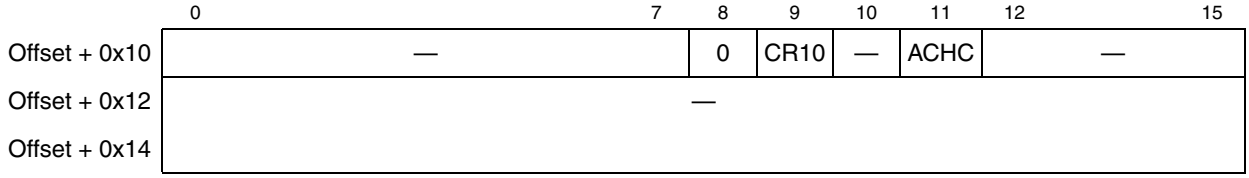


Figure 41-34. AAL0 Protocol-Specific TCT

Table 41-23 describes AAL0 protocol-specific TCT fields.

Table 41-23. AAL0-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0–7	—	Reserved, should be cleared.
	8	0	Must be 0.
	9	CR10	CRC-10 0 CRC10 insertion is disabled. 1 CRC10 insertion is enabled.
	10	—	Reserved, should be cleared.
	11	ACHC	ATM cell header change 0 Normal operation ATM cell header is taken from AAL0 buffer. 1 VPI/VCI (28 bits) are taken from TCT.
	12–15	—	Reserved, should be cleared.
0x12–0x14	—	—	Reserved, should be cleared.

41.10.2.3.4 AAL2 Protocol-Specific TCT

Refer to Section 42.3.5.1, “AAL2 Protocol-Specific TCT.”

41.10.2.3.5 VBR Protocol-Specific TCTE

Figure 41-35 shows the VBR protocol-specific TCTE.

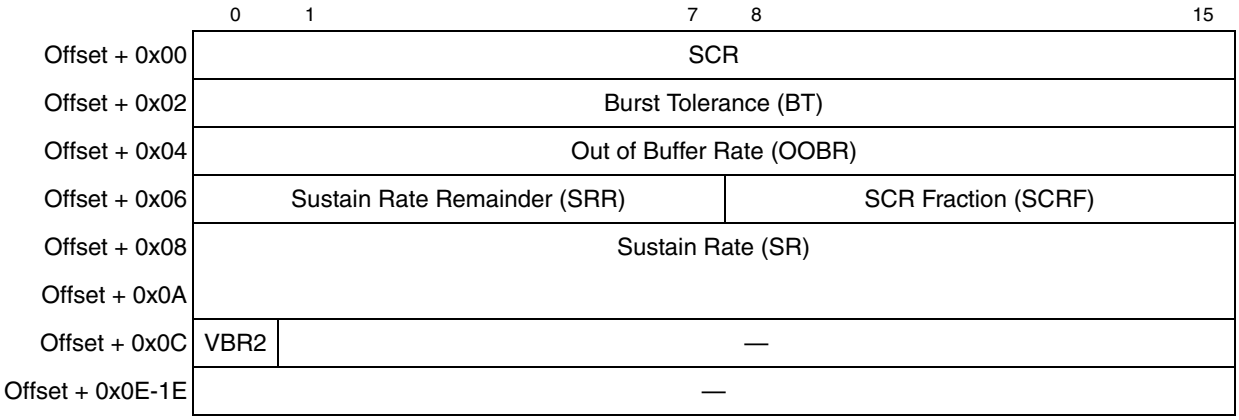


Figure 41-35. Transmit Connection Table Extension (TCTE)—VBR Protocol-Specific

Table 41-24 describes VBR protocol-specific TCTE fields.

Table 41-24. VBR-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	SCR	Sustain cell rate. Holds the sustain cell rate (in slots) permitted for this channel according to the traffic contract. To pace the channel's sustain cell rate, the APC performs a continuous-state leaky bucket algorithm (GCRA).
0x02	—	BT	Burst tolerance. Holds the burst tolerance permitted for this channel according to the traffic contract. The relationship between the BT and the maximum burst size (MBS) is BT=(MBS-2) × (SCR-PCR) + SCR.
0x04	—	OOBR	Out-of-buffer rate. In out of buffer state (when the transmitter tries to open TxBD whose R bit is not set) the APC reschedules the current channel according to OOBR rate.
0x06	0–7	SRR	Sustain rate remainder. Holds the sustain rate remainder after adding the pace fraction field to the additive channel sustain rate. Used by the APC to calculate the channel GCRA (leaky bucket) state. Initialized to 0.
	8–15	SCRF	Holds the sustain cell rate fraction of this channel in units of 1/256 slot.
0x08	—	SR	Sustain rate. Used by the APC to hold the sustain rate after adding the pace field to the additive channel sustain rate. Used by the APC to calculate the channel GCRA (leaky bucket) state.
0x0C	0	VBR2	VBR type 0 Regular VBR. CLP=0+1 cells are rescheduled by PCR or SCR according to the GCRA state. 1 VBR Type 2. CLP=0 cells are rescheduled by PCR or SCR according to the GCRA state. CLP = 1 cells are rescheduled by PCR.
	1–15	—	Reserved, should be cleared.
0x0E–0x1E	—	—	Reserved, should be cleared.

41.10.2.3.6 UBR+ Protocol-Specific TCTE

Figure 41-36 shows the UBR+ protocol-specific TCTE.

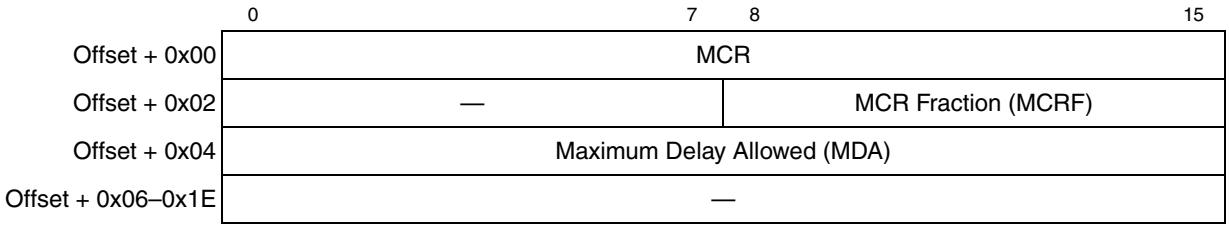


Figure 41-36. UBR+ Protocol-Specific TCTE

Table 41-25 describes UBR+ protocol-specific TCTE fields.

Table 41-25. UBR+ Protocol-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	MCR	Minimum cell rate for this channel. MCR is in units of APC time slots.
0x02	0–7	—	Reserved, should be cleared.
	8–15	MCRF	Minimum cell rate fraction. Holds the minimum cell rate fraction of this channel in units of 1/256 slot.
0x04	—	MDA	Maximum delay allowed. The maximum time-slot service delay allowed for this priority level before the APC reduces the scheduling rate from PCR to MCR.
0x06–0x1E	—	—	Reserved, should be cleared.

41.10.2.3.7 ABR Protocol-Specific TCTE

Figure 41-37 shows the ABR protocol-specific TCTE.

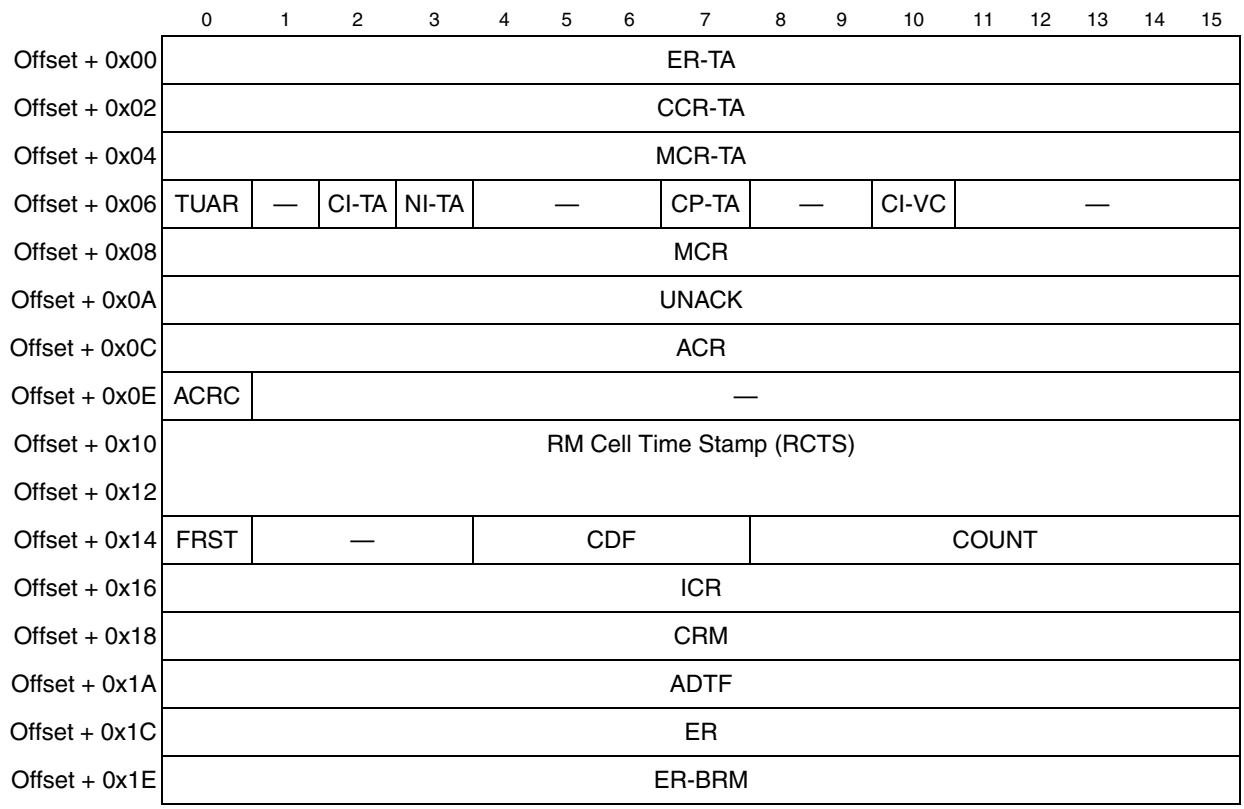


Figure 41-37. ABR Protocol-Specific TCTE

Table 41-26 describes ABR-specific TCTE fields.

Table 41-26. ABR-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	ER-TA	Explicit rate–turn-around cell. Holds the ER of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, this field is overwritten by the new RM cell's ER.
0x02	—	CCR-TA	Current cell rate–turn-around cell. Holds the CCR of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, this field is overwritten by the new RM cell's CCR.
0x04	—	MCR-TA	Minimum cell rate–turn-around cell. Holds the MCR of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell is turned around, this field is overwritten by the new RM cell's MCR.

Table 41-26. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
0x06	0	TUAR	Turn-around flag. The CP sets TUAR to indicate that a new F-RM cell was received, which causes the transmitter to send a B-RM cell whenever the ABR flow control permits. Should be cleared initially.
	1	—	Reserved, should be cleared.
	2	CI-TA	Congestion indication–turn-around cell. Holds the CI of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, CI-TA is overwritten by the new RM cell's CI.
	3	NI-TA	No increase–turn-around cell. Holds the NI of the last received F-RM cell. If another F-RM cell arrives before the previous one was turned around, NI-TA is overwritten by the new RM cell's NI.
	4–6	—	Reserved, should be cleared.
	7	CP-TA	Cell loss priority–turn-around cell. Holds the CLP of the last received F-RM cell. If another F-RM cell arrives before the previous one was turned around, CP-TA is overwritten by the new RM cell's CLP.
	8–9	—	Reserved, should be cleared.
	10	CI-VC	Congestion indication -VC. Holds the EFCI (explicit forward congestion indication) of the last user data cell. The CI bit of the turned around RM cell is ORed with the CI-VC. Should be cleared initially.
	11–15	—	Reserved, should be cleared.
0x08	—	MCR	Minimum cell rate Holds the minimum number of cells/sec of the current ABR channel. Uses the ATMF TM 4.0 floating-point format.
0x0A	—	UNACK	Used by the CP to count F-RM cells sent in an absence of received B-RM cells. Should be cleared initially.
0x0C	—	ACR	Allowed cell rate The cells per second allowed for the current ABR channel. Uses the ATMF TM 4.0 floating-point format. Initialize with ICR.
0x0E	0	ACRC	ACR change. Indicates a change in ACR. Initialize to one.
	1–15	—	Reserved, should be cleared.
0x10	—	RCTS	RM cell time stamp. Used exclusively by the CP. Initialize to zero.
0x14	0	FRST	First turn. Used exclusively by the CP. Indicates the first turn of a backward RM cell, which has priority over a data cell. Initialized to 0.
	1–3	—	Reserved, should be cleared.
	4–7	CDF	Cutoff decrease factor. Controls the decrease in the ACR associated with missing B-RM cells feedback. CDF represents a negative exponent of two, that is, the cutoff decrease factor = 2^{-CDF}. The cutoff decrease factor ranges from 1/64 (CDF = 0b0110) to 1 (CDF = 0b0000). All other CDF values falling outside this range are invalid.
	8–15	COUNT	Count. Used only by the CP. Holds the number of cells sent since the last forward RM cell. Initialize with Nrm (in the parameter RAM).
0x16	—	ICR	Initial cell rate. The number of cells per second of the current ABR channel. The channel's ACR is initialized with ICR. ICR uses the ATMF TM 4.0 floating-point format.
0x18	—	CRM	Missing RM cells count. Limits the number of forward RM cells that may be sent in the absence of received backward RM cell. The CRM is in units of cells.

Table 41-26. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
0x1A	—	ADTF	ADTF–ACR decrease time factor. The ADTF period is 500 ms as defined in the TM 4.0. The ADTF value is defined by the system clock and the time stamp timer prescaler; see Section 21.2.7, "RISC Time-Stamp Control Register (RTSCR)." For a time stamp prescaler of 1 µs, ADTF should be programmed to 500m/(1µs × 1024) = 488.
0x1C	—	ER	Explicit rate. Holds the explicit rate value (in cells/sec) of the current ABR channel. ER is copied to the F-RM cell ER field. The user usually initializes this field to PCR. ER uses the ATMF TM 4.0 floating-point format.
0x1E	—	ER-BRM	Explicit rate-backward RM cell. Holds the maximum explicit rate value (in cells/sec) allowed for B-RM cells. The ER-TA field which is inserted to each B-RM cell is limited by this value. ER-BRM uses the ATMF TM 4.0 floating-point format.

41.10.3 OAM Performance Monitoring Tables

The OAM performance monitoring tables include performance monitoring block test parameters, as shown in Figure 41-38. Each block test needs a 32-byte performance monitoring table in the dual-port RAM. In the connection's RCT and TCT, the user allocates an OAM performance table to a VCC or VPC. See Section 41.6.6, "Performance Monitoring." PMT_BASE in the parameter RAM points to the base address of the tables. The starting address of each PM table is given by PMT_BASE + RCT/TCT[PMT] × 32.

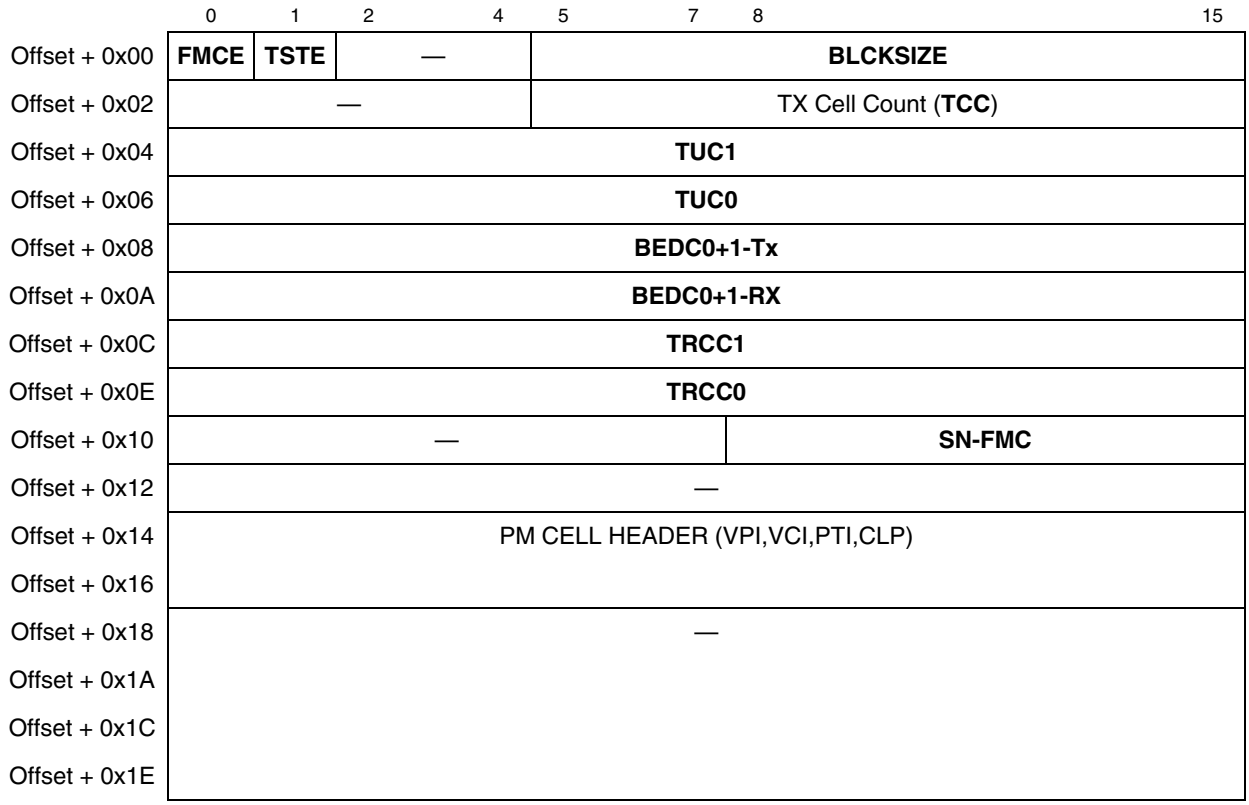


Figure 41-38. OAM Performance Monitoring Table

Table 41-27 describes fields in the performance monitoring table.

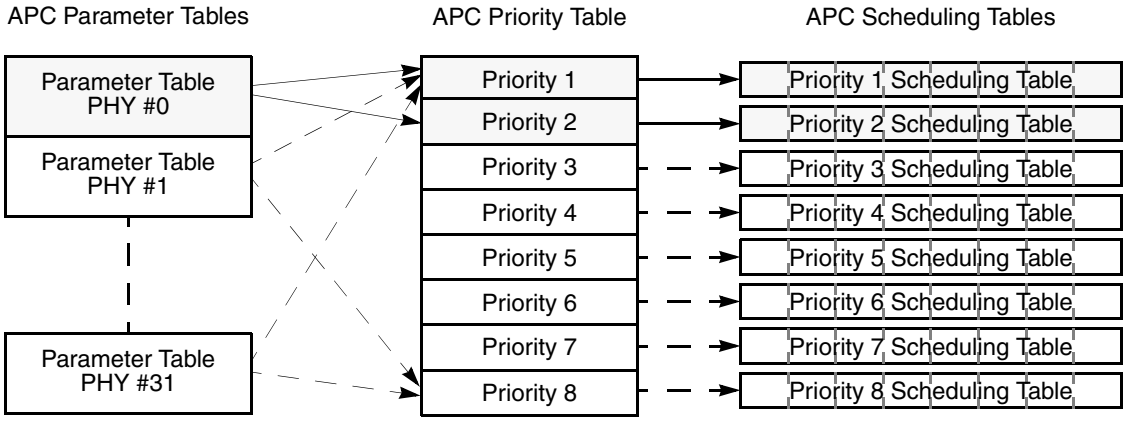
Table 41-27. OAM—Performance Monitoring Table Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**FMCE**	Enables FMC transmission. Initialize to 1.
	1	**TSTE**	FMC time stamp enable 0 The time stamp field of the FMC is coded with all 1s. 1 The value of the time stamp timer is inserted into the time stamp field of the FMC.
	2–4	—	Reserved, should be cleared.
	5–15	**BLCKSIZE**	Performance monitoring block size ranging from 1 to 2,047 cells.
0x02	0–4	—	Reserved, should be cleared.
	5–15	**TCC**	TX cell count. Used by the CP to count data cells sent. Initialize to zero.
0x04	—	**TUC1**	Total user cell 1. Count of CLP = 1 user cells (modulo 65,536) sent. Should be cleared initially.
0x06	—	**TUC0**	Total user cell 0. Count of CLP = 0 user cells (modulo 65,536) sent. Should be cleared initially.
0x08	—	**BEDC0+1-Tx**	Block error detection code 0+1–transmitted cells. Even parity over the payload of the block of user cells sent since the last FMC. Should be cleared initially.
0x0A	—	**BEDC0+1-RX**	Block error detection code 0+1–received cells. Even parity over the payload of the block of user cells received since the last FMC. Should be cleared initially.
0x0C	—	**TRCC1**	Total received cell 1. Count of CLP = 1 user cells (modulo 65,536) received. Should be cleared initially.
0x0E	—	**TRCC0**	Total received cell 0. Count of CLP = 0 user cells (modulo 65,536) received. Should be cleared initially.
0x10	0–7	—	Reserved, should be cleared.
	8–15	**SN-FMC**	Sequence number of the last FMC sent. Should be cleared initially.
0x12	—	—	Reserved, should be cleared.
0x14	—	**PMCH**	PM cell header. Holds the ATM cell header of the FMC, BRC to be inserted by the CP into the Tx cell flow.
0x18–0x1E	—	—	Reserved, should be cleared.

[1] **Boldfaced** entries must be initialized by the user.

41.10.4 APC Data Structure

The APC data structure consists of three elements: the APC parameter tables for the PHY devices, the APC priority table, and the APC scheduling tables. See Figure 41-39.



Note: The shaded areas represent the active structures for an example implementation of PHY #0 with two priorities. (The unshaded areas and dashed arrows represent unused structures.)

Figure 41-39. ATM Pace Control Data Structure

41.10.4.1 APC Parameter Tables

Each PHY's APC parameter table, shown in Table 41-28, holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The table resides in the dual-port RAM. The parameter APCP_BASE, described in Section 41.10.1, "Parameter RAM," points to the base address of PHY#0's parameter table.

For multiple PHYs, the table structure is duplicated. Each table resides in 32 bytes of memory. The starting address of each APC parameter table is given by APCP_BASE + PHY# × 32. Note, however, that in slave mode with multiple PHYs, the parameter table always resides at APCP_BASE regardless of the PHY address.

Table 41-28. APC Parameter Table

Offset[1]	Name	Width	Description
0x00	**APCL_FIRST**	Hword	Address of first entry in the priority table. Must be 8-byte aligned. User-initialized.
0x02	**APCL_LAST**	Hword	Address of last entry in the priority table. Must be 8-byte aligned. User-initialized as APCL_FIRST + 8 x (number_of_priorities - 1).
0x04	**APCL_PTR**	Hword	Address of current priority entry used by the CP. User-initialized with APCL_FIRST.
0x06	**CPS**	Byte	Cells per slot. Determines the number of cells sent per APC slot. See Section 41.3.2, "APC Unit Scheduling Mechanism." User-defined. (0x01 = 1 cell; 0xFF = 255 cells.) **Note:** If ABR is used, CPS must be a power of two.
0x07	**CPS_CNT**	Byte	Cells sent per APC slot counter. User-initialized to CPS; used by the CP.
0x08	**MAX_ITERATION**	Byte	Max iteration allowed. Number of scan iterations allowed in the APC. User-defined. This parameter limits the time spent in a single APC routine, thereby avoiding excessive APC latency.
0x09	**CPS_ABR**	Byte	ABR only. Cells per slot represented as a power of two. User-defined. (For example, if CPS is 1, CPS_ABR = 0x00; if CPS is 8, CPS_ABR = 0x03.)
0x0A	**LINE_RATE_ABR**	Hword	ABR only. The PHY line rate in cells/sec, represented in TM 4.0 floating-point format. User-defined.

Table 41-28. APC Parameter Table (continued)

Offset[1]	Name	Width	Description
0xC	**REAL_TSTP**	Word	Real-time stamp pointer used internally by the APC. Should be cleared initially.
0x10	**APC_STATE**	Word	Used internally by the APC. Should be cleared initially.

[1] Offset values are to APCP_BASE+PHY# × 32. However, in slave mode, the offset is from APCP_BASE regardless of the PHY address.

41.10.4.2 APC Priority Table

Each PHY's APC priority table holds pointers to the APC scheduling table of each priority level. It resides in the dual-port RAM. The priority table can hold up to eight priority levels. Table 41-29 shows the structure of a priority table entry.

Table 41-29. APC Priority Table Entry

Offset	Name	Width	Description
0x00	APC_LEVi_BASE	Hword	APC level i base address. Pointer to the first slot in the APC scheduling table for level i. Should be half-word aligned. User-defined.
0x02	APC_LEVi_END	Hword	APC level i end address. Pointer to the last slot in the APC scheduling table for level i. Should be half-word aligned. User-defined.
0x04	APC_LEVi_RPTR	Hword	APC level i real-time/service pointers. APC table pointers used internally by the APC. Initialize both pointers to APC_LEVi_BASE.
0x06	APC_LEVi_SPTR	Hword	

41.10.4.3 APC Scheduling Tables

The APC uses APC scheduling tables (one table for each priority level) to schedule channel transmission. A scheduling table is divided into time slots, as shown in Figure 41-40. Each slot is a half-word entry. Note that the APC scheduling tables should be cleared before the APC unit is enabled.

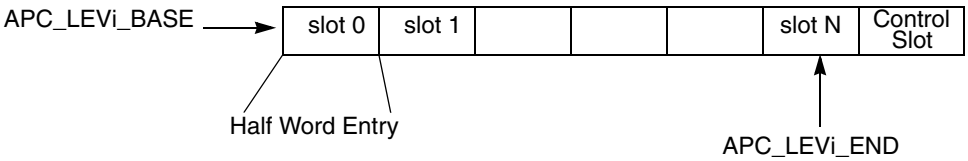


Figure 41-40. APC Scheduling Table Structure

Slot N+1 is used as a control slot, as shown in Figure 41-41.

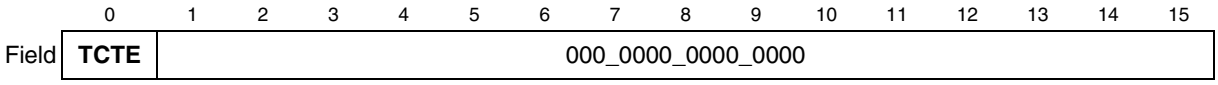


Figure 41-41. Control Slot

Table 41-30 describes control slot fields.

Table 41-30. Control Slot Field Descriptions

Bits	Name[1]	Description
0	**TCTE**	Used for external channels only. 0 Channels in this scheduling table do not use external TCTE. (No external VBR, ABR, UBR+ channels) 1 Channels in this scheduling table use external TCTE. (External VBR, ABR, UBR+ channels)
1–15	—	Reserved, should be cleared.

1 **Boldfaced** entries must be initialized by the user.

41.10.5 ATM Controller Buffer Descriptors (BDs)

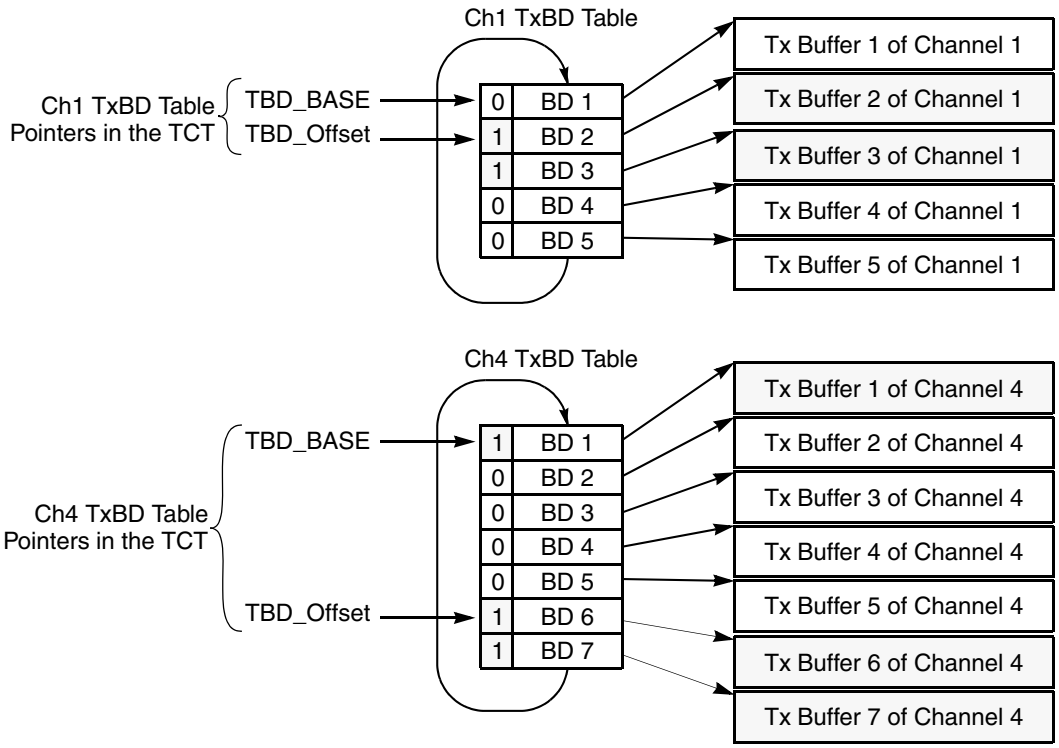
Each ATM channel has separate receive and transmit BD tables. The number of BDs per channel and the size of the buffers is user-defined. The last BD in each table holds a wrap indication. Each BD in the TxBD table points to a buffer to send. At the receive side, the user can choose one of two modes:

- Static buffer allocation. In this mode, the user allocates dedicated buffers to each ATM channel (that is, the user associates each BD with one buffer). Static buffer allocation is useful when the connection rate is known and constant and when data must be reassembled in a particular memory space.
- Global buffer allocation. Available for AAL5 only. In this mode, buffer allocation is dynamic. The user allocates receive buffers and places them in global buffer pools. When the CP needs a receive buffer, it first fetches a buffer pointer from one of the global buffer pools and writes the pointer to the current RxBD. Global buffer allocation is optimized for allocating memory among many ATM channels with variable data rates, such as ABR channels.

41.10.5.1 Transmit Buffer Operation

The user prepares a table of BDs pointing to the buffers to be sent. The address of the first BD is put in the channel's TCT[TBD_BASE]. The transmit process starts when the core issues an ATM TRANSMIT command. The CP reads the first TxBD in the table and sends its associated buffer. When the current buffer is finished, the CP increments TBD_Offset, which holds the offset from TBD_BASE to the current BD. It then reads the next BD in the table. If the BD is ready (TxBD[R] = 1), the CP continues sending. If the current BD is not ready, the CP polls the ready bit at the channel rate unless TCT[AVCF] = 1, in which case the CP removes the channel from the APC and clears TCT[VCON]. The core must issue a new ATM TRANSMIT command to restart transmission.

Figure 41-42 shows the ready bit in the TxBD tables and their associated buffers for two example ATM channels.



Note: The shaded buffers are ready to be sent; unshaded buffers are waiting to be prepared.

Figure 41-42. Transmit Buffers and BD Table Example

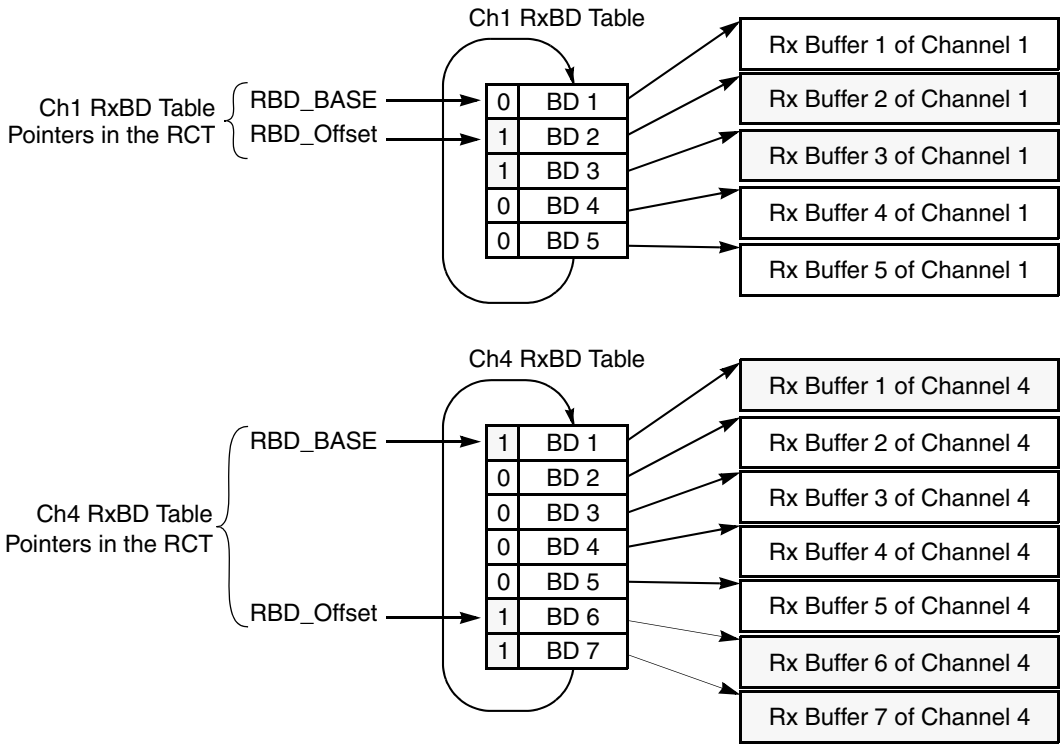
41.10.5.2 Receive Buffer Operation

For AAL5 channels, the user should choose to operate in static buffer allocation or in global buffer allocation by writing to RCT[BUFM]. AAL0 channels must use static buffer allocation.

41.10.5.2.1 Static Buffer Allocation

The user prepares a table of BDs pointing to the receive buffers. The address of the first BD is put in the channel's RCT[RBD_BASE]. When an ATM cell arrives, the CP opens the first BD in the table and starts filling its associated buffer with received data. When the current buffer is full, the CP increments RBD_Offset, which is the offset to the current BD from RBD_BASE, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP continues receiving. If the BD is not empty, a busy condition has occurred and a busy interrupt is sent to the event queue.

Figure 41-43 shows the empty bit in the RxBD tables and their associated buffers for two example ATM channels.



Note: The shaded buffers are empty; unshaded buffers are waiting to be processed.

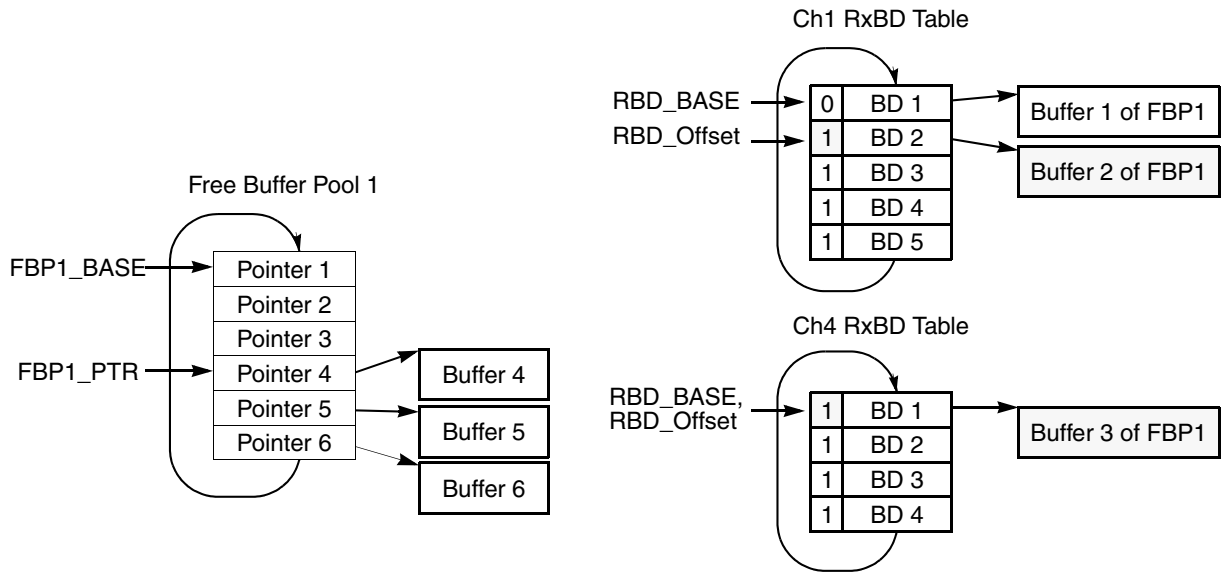
Figure 41-43. Receive Static Buffer Allocation Example

41.10.5.2.2 Global Buffer Allocation

The user prepares a table of BDs without assigning buffers to them (no buffer pointers). The address of the first BD is put into the channel's RCT[RBD_BASE]. The user also prepares sets of free buffers (of size RCT[MRBLR]) in up to four free buffer pools (chosen in RCT[BPOOL]); see Section 41.10.5.2.3, "Free Buffer Pools."

When an ATM cell arrives, the CP opens the first BD in the table, fetches a buffer pointer from the free buffer pool associated with this channel, and writes the pointer to RxBD[RXDBPTR], the receive data buffer pointer field in the BD. When the current buffer is full, the CP increments RBD_Offset, which is the offset from the RBD_BASE to the current BD, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP fetches another buffer pointer from the free buffer pool and reception continues. If the BD is not empty, a busy condition occurs and a busy interrupt is sent to the event queue specifying the ATM channel code. As software then processes each full buffer (RxBD[E] = 0), it sets RxBD[E] and copies the buffer pointer back to the free buffer pool.

Figure 41-44 shows two ATM channels' BD tables and one free buffer pool. Both channels are associated with free buffer pool 1. The CP allocates the first two buffers of buffer pool 1 to channel 1 and the third to channel 4.



Note: Buffers 2 and 3 are receiving data. After buffer 1 is processed, it can be returned to the pool.

Figure 41-44. Receive Global Buffer Allocation Example

41.10.5.2.3 Free Buffer Pools

As Figure 41-45 shows, when a buffer pointer is fetched from a pool, the CP clears the entry's valid bit and increments FBP#_PTR. After the CP uses an entry with the wrap bit set (W = 1), it returns to the first entry in the pool. After a buffer pointer is returned to the pool, the user should set V to indicate that the entry is valid. If the CP tries to read an invalid entry (V = 0), the buffer pool is out of free buffers; the global-buffer-pool-busy event is then set in FCCE[GBPB] and a busy interrupt is sent to the interrupt queue specifying the ATM channel code associated with the pool.

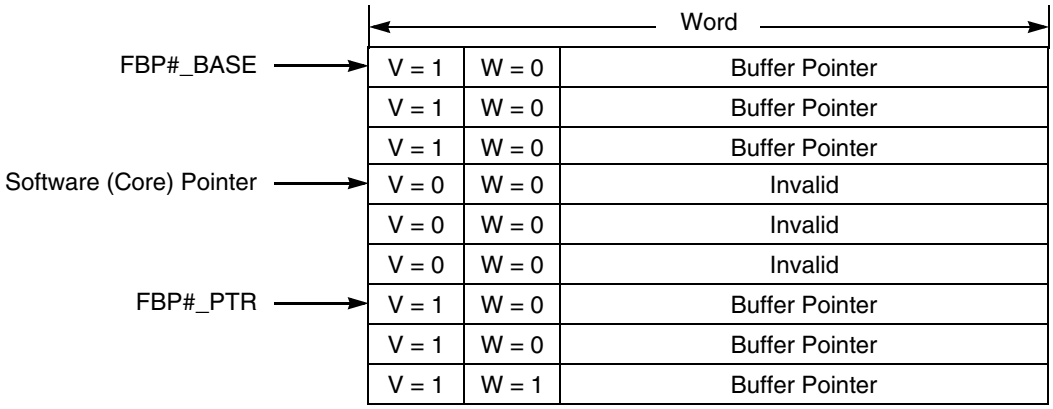


Figure 41-45. Free Buffer Pool Structure

Figure 41-46 describes the structure of a free buffer pool entry.

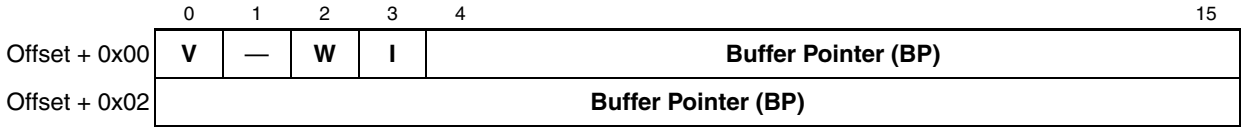


Figure 41-46. Free Buffer Pool Entry

Table 41-31 describes free buffer pool entry fields.

Table 41-31. Free Buffer Pool Entry Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**V**	Valid buffer entry. 0 This free buffer pool entry contains an invalid buffer pointer. 1 This free buffer pool entry contains a valid buffer pointer.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap bit. When set, this bit indicates the last entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3	**I**	Red-line interrupt. Can be used to indicate that the free buffer pool has reached a red line and additional buffers should be added to this pool to avoid a busy condition. 0 No interrupt is generated. 1 A red-line interrupt is generated when this buffer is fetched from the free buffer pool.
	4–15	**BP**	Buffer pointer. Points to the start address of the receive buffer. The four msbs are control bits, and the four msbs of the real buffer pointer are taken from the four msbs of the parameter FBP_ENTRY_EXT in the free buffer pool parameter table.
0x02	0–15		

[1] **Boldfaced** entries must be initialized by the user.

41.10.5.2.4 Free Buffer Pool Parameter Tables

The free buffer pool parameters are held in parameter tables in the dual-port RAM; see Table 41-32. FBT_BASE in the parameter RAM points to the base address of these tables. Each of the four free buffer pools has its own parameter table with a starting address given by FBT_BASE+ RCT[BPOOL] × 16.

Table 41-32. Free Buffer Pool Parameter Table

Offset [1]	Bits	Name[2]	Description
0x00	—	**FBP_BASE**	Free buffer pool base. Holds the pointer to the first entry in the free buffer pool. FBP_BASE should be word aligned. User-defined.
0x04	—	**FBP_PTR**	Free buffer pool pointer. Pointer to the current entry in the free buffer pool. Initialize to FBP_BASE.
0x08	—	**FBP_ENTRY_EXT**	Free buffer pool entry extension. FBP_ENTRY_EXT[0–3] holds the four left bits of FBP_ENTRY. FBP_ENTRY_EXT[4–15] should be cleared. User-defined.

Table 41-32. Free Buffer Pool Parameter Table (continued)

Offset [1]	Bits	Name[2]	Description
0x0A	0	**BUSY**	The CP sets this bit when it tries to fetch buffer pointer with V bit clear. FCCE[GBPB] is also set. Initialize to zero.
	1	**RLI**	Red-line interrupt. Set by the CP when it fetches a buffer pointer with I = 1. FCCE[GRLI] is also set. Initialize to zero.
	2–7	—	Reserved, should be cleared.
	8	**EPD**	Early packet discard. 0 Normal operation. 1 AAL5 frames in progress are received, but new AAL5 frames associated with this pool are discarded. Can be used to implement EPD under core control.
	9–15	**—**	Reserved, should be cleared.
0x0C	—	**FBP_ENTRY**	Free buffer pool entry. Initialize with the first entry of the free buffer pool. Note that FBP_ENTRY must be reinitialized with the entry pointed to by FBP_PTR when a busy state occurs to re-enable free buffer pool processing.

1 Offset from FBT_BASE+RCT[BPOOL] × 16.
2 **Boldfaced** entries must be initialized by the user.

41.10.5.3 ATM Controller Buffers

Table 41-33 describes properties of the ATM receive and transmit buffers.

Table 41-33. Receive and Transmit Buffers

AAL	Receive		Transmit	
	Size	Alignment	Size	Alignment
AAL5	Multiple of 48 octets (except last buffer in frame)	Double word aligned	Any	No requirement
AAL1	At least 47 octets	No requirement	At least 47 octets	No requirement
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

41.10.5.4 AAL5 RxBD

Figure 41-47 shows the AAL5 RxBD.

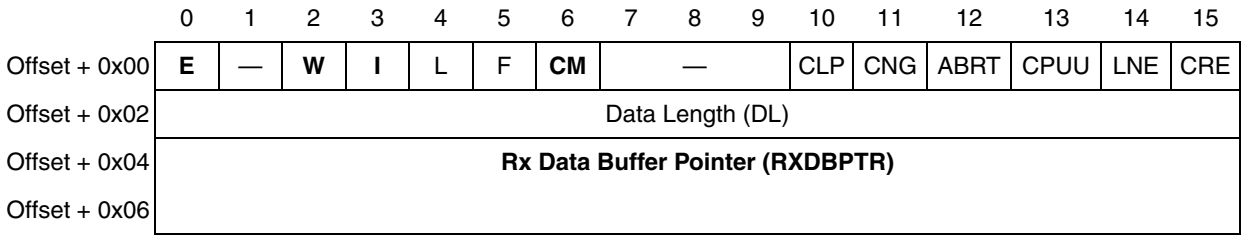


Figure 41-47. AAL5 RxBD

Table 41-34 describes AAL5 RxBD fields.

Table 41-34. AAL5 RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Empty. 0 The buffer associated with this RxBD is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The buffer associated with this RxBD is empty or reception is in progress. This RxBD and its receive buffer are controlled by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set in the event register when INT_CNT reaches the global interrupt threshold.
	4	L	Last in frame. Set by the ATM controller for the last buffer in a frame. 0 Buffer is not last in a frame. 1 Buffer is last in a frame. ATM controller writes frame length in DL and updates the error flags.
	5	F	First in frame. Set by the ATM controller for the first buffer in a frame. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
	6	**CM**	Continuous mode 0 Normal operation. 1 The CP does not clear the empty bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–9	—	Reserved, should be cleared.
	10	CLP	Cell loss priority. At least one cell associated with the current message was received with CLP = 1. May be set at the last buffer of the message.
	11	CNG	Congestion indication. The last cell associated with the current message was received with PTI middle bit set. CNG may be set at the last buffer of the message.
	12	ABRT	Abort message indication. The current message was received with Length field zero.
	13	CPUU	CPCS-UU+CPI indication. Set when the CPCS-UU+CPI field is non zero. CPUU may be set at the last buffer of the message.
	14	LNE	Rx length error. AAL5 CPCS-PDU length violation. May be set only for the last BD of the frame if the pad length is greater than 47 or less than zero octets.
	15	CRE	Rx CRC error. Indicates CRC32 error in the current AAL5 PDU. Set only for the last BD of the frame.

Table 41-34. AAL5 RxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	DL	Data length. The number of octets written by the CP into this BD's buffer. It is written by the CP once the BD is closed. In the last BD of a frame, DL contains the total frame length.
0x04		**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in internal or external memory. This pointer must be burst-aligned.

[1] **Boldfaced** entries must be initialized by the user.

41.10.5.5 AAL1 RxBD

Figure 41-48 shows the AAL1 RxBD.

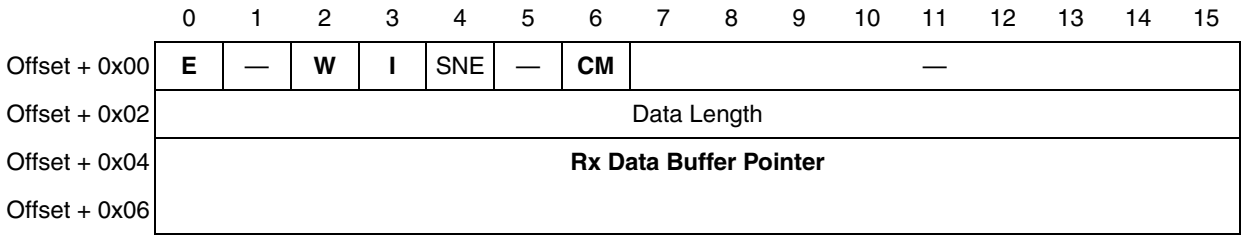


Figure 41-48. AAL1 RxBD

Table 41-35 describes AAL1 RxBD fields.

Table 41-35. AAL1 RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Empty 0 The buffer associated with this RxBD is filled with received data or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP cannot use this BD again while E = 0. 1 The buffer is not full. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer is used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table overall space is constrained to 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4	SNE	Sequence number error. SNE is set when a sequence number error is detected in the current AAL1 CES buffer.
	5	—	Reserved, should be cleared.

Table 41-35. AAL1 RxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x00	6	**CM**	Continuous mode 0 Normal operation. 1 The empty bit (RxBD[E]) is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–15	—	Reserved, should be cleared.
0x02	—	DL	Data length. The number of octets the CP writes into the buffer once its BD is closed.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

[1] **Boldfaced** entries must be initialized by the user.

41.10.5.6 AAL0 RxBD

Figure 41-49 shows the AAL0 RxBD.

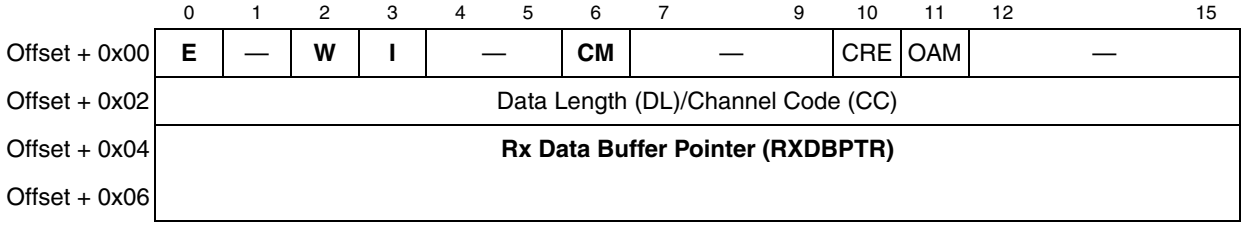


Figure 41-49. AAL0 RxBD

Table 41-36 describes AAL0 RxBD fields.

Table 41-36. AAL0 RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Empty 0 The buffer associated with this RxBD is filled with received data, or data reception was aborted due to an error. The core can examine or write to any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The Rx buffer is empty or reception is in progress. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of the current channel. After this buffer has been used, the CP will receive incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.

Table 41-36. AAL0 RxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x00	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation. 1 The CP does not clear the E bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–9	—	Reserved, should be cleared.
	10	CRE	Rx CRC error. Indicates a CRC10 error in the current AAL0 buffer. The CRE bit is considered an error only if the received cell had a CRC10 field in the cell payload.
	11	OAM	Operation and maintenance cell. If OAM is set, the current AAL0 buffer contains an OAM cell. This cell is associated with the channel indicated by the channel code field (CC field).
	12-15	—	Reserved, should be cleared.
0x02	—	DL/CC	Data length/channel code. If RxBD[OAM] is set, this field functions as CC; otherwise, it is DL. Data length is the size in octets of this buffer (MRBLR value). Channel code specifies the channel code associated with this OAM cell.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

1 **Boldfaced** entries must be initialized by the user.

41.10.5.7 AAL2 RxBD

Refer to Section 42.4.4.4, “CPS Receive Buffer Descriptor (RxBD).”

41.10.5.8 AAL5 User-Defined Cell—RxBD Extension

In user-defined cell mode, the AAL5 and AAL1 CES RxBDs are extended to 32 bytes; see Figure 41-50.

NOTE

For AAL0, a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

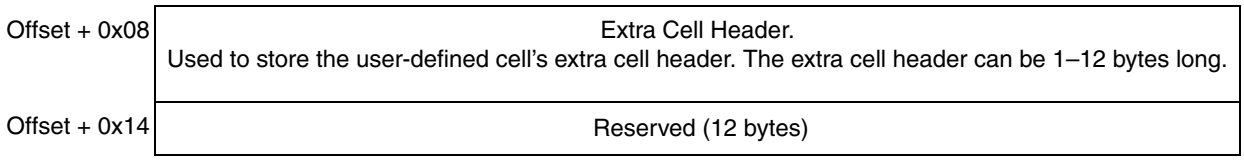


Figure 41-50. User-Defined Cell—RxBD Extension

41.10.5.9 AAL5 TxBDs

Figure 41-51 shows the AAL5 TxBD.

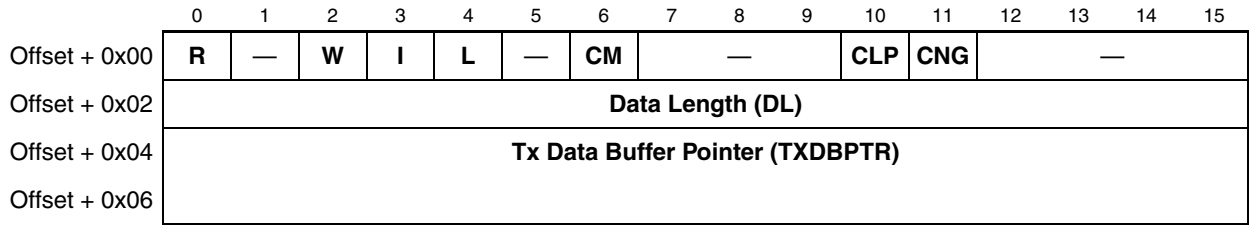


Figure 41-51. AAL5 TxBD

Table 41-37 describes AAL5 TxBD fields.

Table 41-37. AAL5 TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4	**L**	Last in frame. Set by the user to indicate the last buffer in a frame. 0 Buffer is not last in a frame. 1 Buffer is last in a frame.
	5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of CM.
	7-9	—	Reserved, should be cleared.
	10	**CLP**	The ATM cell header CLP bit of the cells associated with the current frame are ORed with this field. This field is valid only in the first BD of the frame.
	11	**CNG**	The ATM cell header CNG bit of the cells associated with the current frame are ORed with this field. This field is valid only in the first BD of the frame.
	12–15	—	Reserved, should be cleared.

Table 41-37. AAL5 TxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	**DL**	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer, which may or may not be 8-byte-aligned. The buffer may reside in either internal or external memory. This value is not modified by the CP.

1 **Boldfaced** entries must be initialized by the user.

41.10.5.10 AAL1 TxBDs

Figure 41-52 shows the AAL1 TxBD.

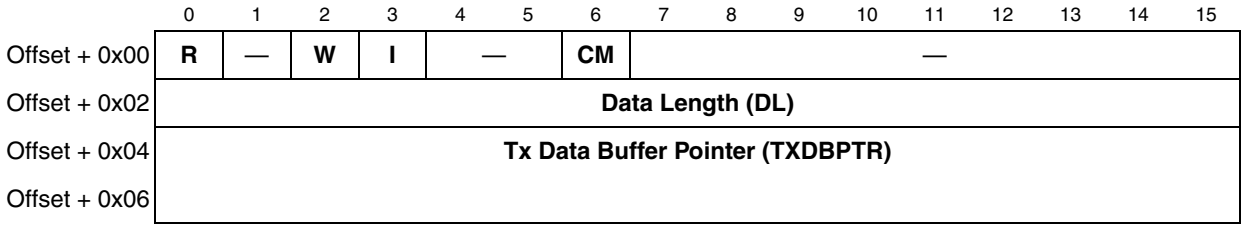


Figure 41-52. AAL1 TxBD

Table 41-38 describes AAL1 TxBD fields.

Table 41-38. AAL1 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears this bit after the buffer has been sent or after an error condition is encountered. 1 The buffer prepared for transmission by the user has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.

Table 41-38. AAL1 TxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x00	7–11	—	Reserved, should be cleared.
0x02	—	**DL**	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. The buffer may reside in either internal or external memory. This value is not modified by the CP.

41.10.5.11 AAL0 TxBDs

Figure 41-53 shows AAL0 TxBDs. Note that the data length field is calculated internally as 52 bytes, plus the extra header length (defined in FPSMR[TEHS]) when in UDC mode.

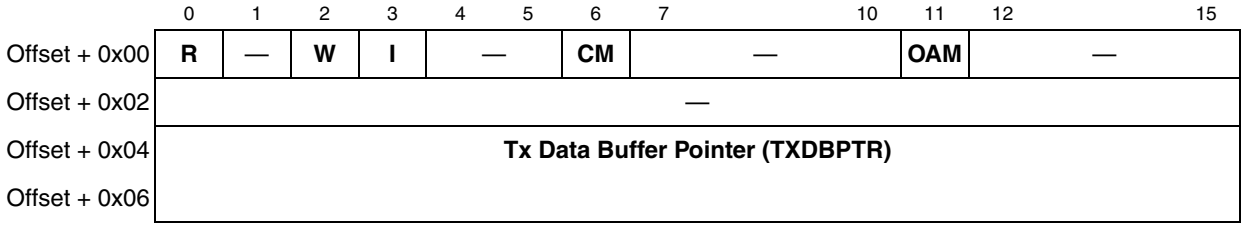


Figure 41-53. AAL0 TxBDs

Table 41-39 describes AAL0 TxBD fields.

Table 41-39. AAL0 TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**R**	Ready 0 The buffer is not ready for transmission. The user can manipulate this BD or its buffer. The CP clears R after the buffer has been sent or after an error occurs. 1 The buffer that the user prepared for transmission has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined by the W bit. The current table is constrained to 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.
	7–10	—	Reserved, should be cleared.

Table 41-39. AAL0 TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	11	**OAM**	Operation and maintenance cell. If OAM is set, the current AAL0 buffer contains an F5 or F4 OAM cell. Performance monitoring calculations are not done on OAM cells.
	11–15	—	Reserved, should be cleared.
0x02	—	—	Reserved, should be cleared.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer, which may or may not be 8-byte-aligned. The buffer may reside in either internal or external memory. This value is not modified by the CP.

1 **Boldfaced** entries must be initialized by the user.

41.10.5.12 AAL2 TxBDs

Refer to Section 42.3.5.5, "SSSAR Transmit Buffer Descriptor."

41.10.5.13 AAL5, AAL1 User-Defined Cell—TxBD Extension

In user-defined cell mode, the AAL5 and AAL1 TxBDs are extended to 32 bytes; see Figure 41-54.

NOTE

For AAL0 a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

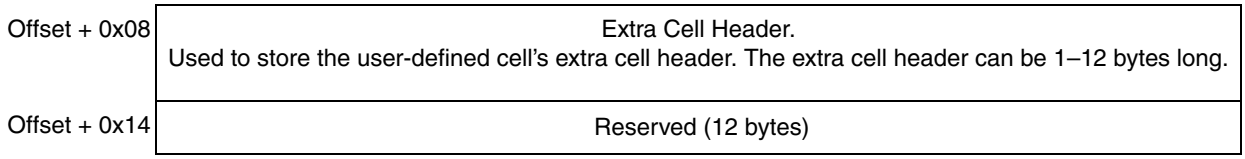


Figure 41-54. User-Defined Cell—TxBD Extension

41.10.6 AAL1 Sequence Number (SN) Protection Table

The 32-byte sequence number protection table, pointed to by AAL1_SNPT_BASE in the ATM parameter RAM, resides in dual-port RAM and is used for AAL1 only. The table should be initialized according to Figure 41-55.

	0	15
Offset + 0x00	0x0000	
Offset + 0x02	0x0007	
Offset + 0x04	0x000D	
Offset + 0x06	0x000A	
Offset + 0x08	0x000E	
Offset + 0x0A	0x0009	
Offset + 0x0C	0x0003	
Offset + 0x0E	0x0004	
Offset + 0x10	0x000B	
Offset + 0x12	0x000C	
Offset + 0x14	0x0006	
Offset + 0x16	0x0001	
Offset + 0x18	0x0005	
Offset + 0x1A	0x0002	
Offset + 0x1C	0x0008	
Offset + 0x1E	0x000F	

Figure 41-55. AAL1 Sequence Number (SN) Protection Table

41.10.7 UNI Statistics Table

The UNI statistics table, shown in Table 41-40, resides in the dual-port RAM and holds UNI statistics parameters. UNI_STATT_BASE points to the base address of this table. Each PHY's own table has a starting address given by UNI_STATT_BASE+ PHY# × 8.

Table 41-40. UNI Statistics Table

Offset[1]	Name	Width	Description
0x00	UTOPIAE	Hword	Counts cells dropped as a result of UTOPIA/ATM protocol violations. Violations include the following: 1. Parity error 2. HEC error 3. Invalid timing of RxSOC. If RxClav is asserted for the selected PHY, RxSOC should be asserted the cycle immediately following the assertion of $\overline{\text{RXENB}}$. A violation occurs if RxSOC is not asserted at that time (that is, it is late or missing).
0x02	MIC_COUNT	Hword	Counts misinserted cells dropped as a result of address look-up failure.
0x04	CRC10E_COUNT	Hword	Counts cells dropped as a result of CRC10 failure. AAL5-ABR only.
0x06	—	Hword	Reserved, should be cleared.

[1] Offset from UNI_STATT_BASE+PHY# × 8.

41.11 ATM Exceptions

The ATM controller interrupt handling involves two principal data structures: FCCEs (FCC event registers) and circular interrupt queues.

Four priority interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ], the user determines which queue receives the interrupt. Channel Rx buffer, Rx frame, or Tx buffer events can be masked by clearing interrupt mask bits in RCT and TCT.

NOTE

Ensure that the transmit INTQs and the receive INTQs are programmed as separate queues.

After an interrupt request, the host reads FCCE. If FCCE[GINT*x*] = 1, at least one entry was added to one of the interrupt queues. After clearing FCCE[GINT*x*], the host processes the valid interrupt queue entries and clears each entry's valid bit. The host follows this procedure until it reaches an entry with V = 0. See Section 41.11.2, "Interrupt Queue Entry."

The host controls the number of interrupts sent to the core using a down counter in the interrupt queue's parameter table; see Section 41.11.3, "Interrupt Queue Parameter Tables." For each event sent to an interrupt queue, a counter (that has been initialized to a threshold number of interrupts) is decremented. When the counter reaches zero, the global interrupt, FCCE[GINT*x*], is set.

41.11.1 Interrupt Queues

Interrupt queues are located in external memory. The parameters of each queue are stored in a table. See Section 41.11.3, "Interrupt Queue Parameter Tables."

When an interrupt occurs, the CP writes a new entry to the interrupt queue, the V bit is set, and the queue pointer (INTQ_PTR) is incremented. Once the CP uses an entry with W = 1, it returns to the first entry in the queue. If the CP tries to overwrite a valid entry (V = 1), an overflow condition occurs and the queue's overflow flag, FCCE[INTO*x*], is set.

The interrupt queue structure is displayed in Figure 41-56.

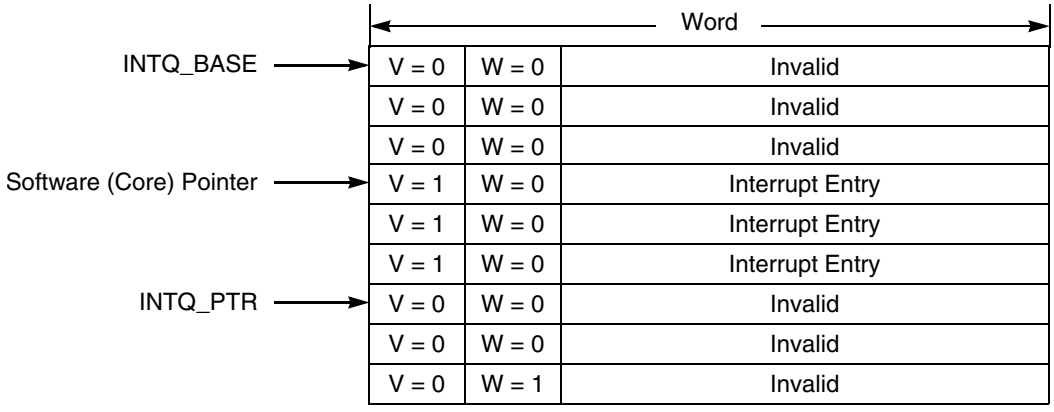


Figure 41-56. Interrupt Queue Structure

41.11.2 Interrupt Queue Entry

Each one-word interrupt queue entry provides detailed interrupt information to the host. Figure 41-57 shows an entry.

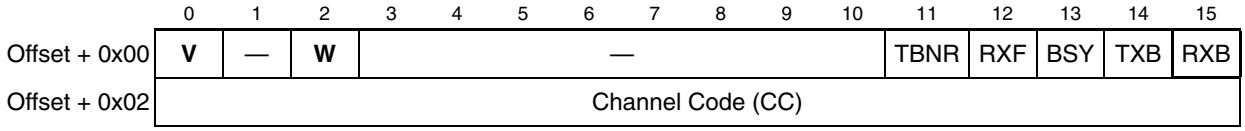


Figure 41-57. Interrupt Queue Entry

Table 41-41 describes interrupt queue entry fields.

Table 41-41. Interrupt Queue Entry Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be use by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	—	Reserved, should be cleared.
	11	TBNR	Tx buffer-not-ready. Set when a transmit buffer-not-ready interrupt is issued. This interrupt is issued when the CP tries to open a TxBD that is not ready (R = 0). This interrupt is sent only if TCT[BNM] = 1. This interrupt has an associated channel code. Note that for AAL5, this interrupt is sent only if frame transmission is started. In this case, an abort frame transmission is sent (last cell with length=0), the channel is taken out of the APC, and the TCT[VCON] flag is cleared.
	12	RXF	Rx frame. RXF is set when an Rx frame interrupt is issued. This interrupt is issued at the end of AAL5 PDU reception. This interrupt is issued only if RCT[RXFM] = 1. This interrupt has an associated channel code.
	13	BSY	Busy condition. The BD table or the free buffer pool associated with this channel is busy. Cells were discarded due to this condition. This interrupt has an associated channel code.
	14	TXB	Tx buffer. TXB is set when a transmit buffer interrupt is issued. This interrupt is enabled when both TxBD[I] and TCT[IMK] = 1. This interrupt has an associated channel code.
	15	RXB	Rx buffer. RXB is set when an Rx buffer interrupt is issued. This interrupt is enabled when both RxBD[I] and RCT[RXBM] = 1. This interrupt has an associated channel code.
0x02	—	CC	Channel code specifies the channel associated with this interrupt.

[1] **Boldfaced** entries must be initialized by the user.

41.11.3 Interrupt Queue Parameter Tables

The interrupt queue parameters are held in parameter tables in the dual-port RAM; see Table 41-42. INTT_BASE in the parameter RAM points to the base address of these tables. Each of the four interrupt queues has its own parameter table with a starting address given by INTT_BASE+ RCT/TCT[INTQ] × 16.

Table 41-42. Interrupt Queue Parameter Table

Offset [1]	Name[2]	Width	Description
0x00	**INTQ_BASE**	Word	Base address of the interrupt queue. User-defined.
0x04	**INTQ_PTR**	Word	Pointer to interrupt queue entry. Initialize to INTQ_BASE.
0x08	**INT_CNT**	Half Word	Interrupt counter. Initialize with INT_ICNT. The CP decrements INT_CNT for each interrupt. When INT_CNT reaches zero, the queue's global interrupt flag FCCE[GINT*x*] is set.
0x0A	**INT_ICNT**	Half Word	Interrupt initial count. User-defined global interrupt threshold—the number of interrupts required before the CP issues a global interrupt (FCCE[GINT*x*]).
0x0C	**INTQ_ENTRY**	Word	Interrupt queue entry. Must be initialized to the entry pointed to by INTQ_PTR, which is initially the first empty entry of the queue. Note that after an overrun occurs, this entry must be reset to the entry pointed to by INTQ_PTR to re-enable interrupt processing.

1 Offset from INTT_BASE+RCT/TCT[INTQ] × 16.

2 **Boldfaced** entries must be initialized by the user.

41.12 UTOPIA Interface

The ATM controller interfaces with a PHY device through the UTOPIA interface. The MPC8555E supports UTOPIA level II for both master and slave modes.

41.12.1 Extended Number of PHYs

The MPC8555E has additional pin muxing to support 31 PHYs on both FCC1 and FCC2. To utilize this feature, do the following:

- Select dedicated UTOPIA address lines for FCC1 and FCC2 in the parallel I/O (TxADDR[4:3], RxADDR[4:3]).

Refer to Chapter 45, "Parallel I/O Ports."

41.12.2 UTOPIA Interface Master Mode

Cell transfer on an ATM device (with single or multiple PHYs) uses cell-level handshaking as defined in the UTOPIA standards. The FCC does not pause cell transmission by the PHY and does not stop receiving cells from the PHY.

UTOPIA master signals are shown in Figure 41-58.

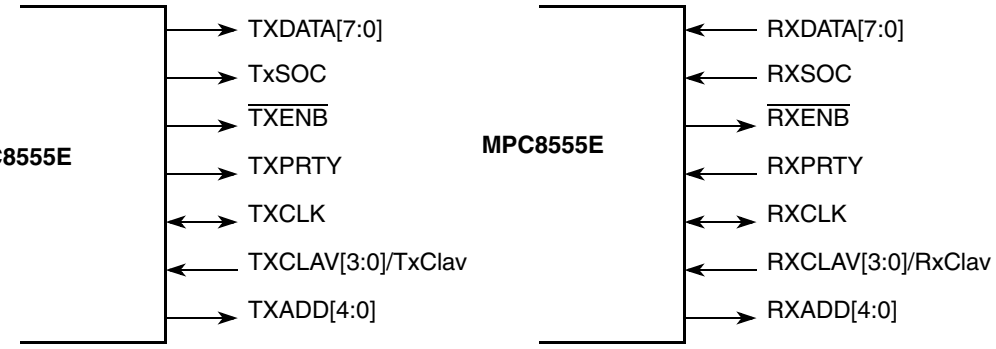


Figure 41-58. UTOPIA Master Mode Signals

Table 41-43 describes UTOPIA master mode signals.

Table 41-43. UTOPIA Master Mode Signal Descriptions

Signal	Description
TxDATA[7:0]	Carries transmit data from the ATM controller to a PHY device. TxDATA[7] is the msb, TxDATA[0] is the lsb.
TxSOC	Transmit start of cell. Asserted by the ATM controller when the first byte of a cell is sent on TxDATA lines.
TxENB	Transmit enable. Asserted by the ATM controller when valid data is placed on the TxDATA lines.
TxClav/TxCLAV[3:0]	Transmit cell available. Asserted by the PHY device to indicate that the PHY has room for a complete cell.
TxPRTY	Transmit parity. Asserted by the ATM controller. It is an odd parity bit over the TxDATA bits.
TxCLK	Transmit clock. Provides the synchronization reference for the TxDATA, TxSOC, $\overline{\text{TxENB}}$, TxCLAV, TxPRTY signals. All the above signals are sampled at low-to-high transitions of TxCLK.
TxADD[4:0]	Transmit address. Address bus from the ATM controller to the PHY device used to select the appropriate M-PHY device. Each M-PHY device needs to maintain its address. TxADD[4] is the msb.
RxDATA[7:0]	Carries receive data from the PHY to the ATM controller. RxDATA[7] is the msb, RxDATA[0] is the lsb.
RxSOC	Receive start of cell. Asserted by the PHY device as the first byte of a cell is received on RxDATA.
RxENB	Receive enable. An ATM controller asserts to indicate that RxDATA and RxSOC will be sampled at the end of the next RxCLK cycle. For multiple PHYs, $\overline{\text{RxENB}}$ is used to three-state RxDATA and RxSOC at each PHY's output. RxDATA and RxSOC should be enabled only in cycles after those with $\overline{\text{RxENB}}$ asserted.
RxClav/RxCLAV[3:0]	Receive cell available. Asserted by a PHY device when it has a complete cell to give the ATM controller.
RxPRTY	Receive parity. Asserted by the PHY device. It is an odd parity bit over the RxDATA. If there is a RxPRTY error and the receive parity check FPSMR[RxP] is cleared, the cell is discarded. See Section 41.13.3, "FCC Protocol-Specific Mode Register (FPSMR)," and Section 41.10.7, "UNI Statistics Table."
RxCLK	Receiver clock. Synchronization reference for RxDATA, RxSOC, $\overline{\text{RxENB}}$, RxCLAV, and RxPRTY, all of which are sampled at low-to-high transitions of RxCLK.
RxADD[4:0]	Receive address. Address bus from the ATM controller to the PHY device used to select the appropriate M-PHY device. Each M-PHY device needs to maintain its address. RxADD[4] is the msb.

41.12.2.1 UTOPIA Master Multiple PHY Operation

The MPC8555E supports two polling modes:

- Direct polling uses CLAV[3:0] with PHY selection using ADD[2:0]. Up to four PHYs can be supported.
- Single CLAV polling uses Clav and ADD[4:0]. ATM controller polls all active PHYs starting from PHY address 0x0 to the address written in FPSMR[LAST_PHY]. Up to 31 PHY devices are supported.

Both modes support round-robin priority or fixed priority, described in Section 41.13.3, "FCC Protocol-Specific Mode Register (FPSMR)."

41.12.3 UTOPIA Interface Slave Mode

In UTOPIA slave mode (single or multiple PHY), cells are transferred using cell-level and octet-level handshakes as defined by the UTOPIA level II standard. The FCC allows cell transfer to be halted or paused. If the master negates $\overline{\text{TXENB}}$, the cell that the FCC is transmitting is halted. If the master negates $\overline{\text{RXENB}}$, the cell that the FCC is receiving is paused. Note the following restriction on halting a cell transfer: there cannot be a halt immediately before the transfer of the last data word. There is no restriction on pausing a cell transfer.

UTOPIA slave signals are shown in Figure 41-59.

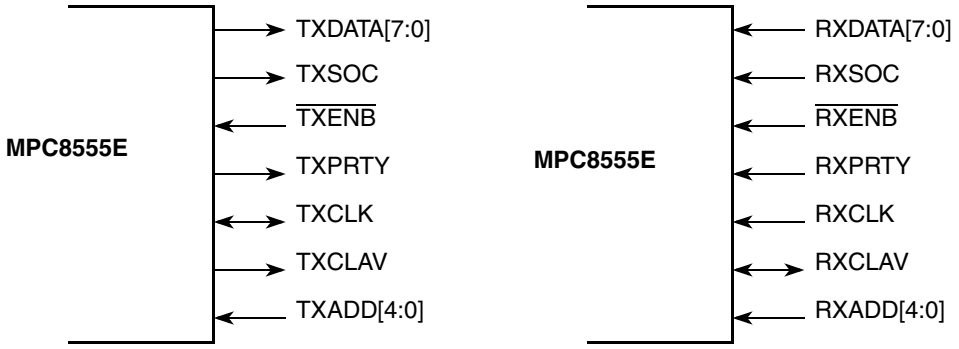


Figure 41-59. UTOPIA Slave Mode Signals

Table 41-44 describes UTOPIA slave mode signals.

Table 41-44. UTOPIA Slave Mode Signals

Signal	Description
TxDATA[7:0]	Transmit data bus. Carries transmit data from the ATM controller to the master device. TxDATA[7] is the msb, TxDATA[0] is the lsb.
TxSOC	Transmit start of cell. Asserted by an ATM controller as the first byte of a cell is sent on the TxDATA lines.
TxENB	Transmit enable. An input to the ATM controller. It is asserted by the UTOPIA master to signal the slave to send data in the next TxCLK cycle.
TxCLAV	Transmit cell available. Asserted by the ATM controller to indicate it has a complete cell to transmit.
TxPRTY	Transmit parity. Asserted by the ATM controller. It is an odd parity bit over the TxDATA.

Table 41-44. UTOPIA Slave Mode Signals (continued)

Signal	Description
TxCLK	Transmit clock. Provides the synchronization reference for the TxDATA, TxSOC, $\overline{\text{TxENB}}$, TxCLAV, and TxPRTY signals. All of the above signals are sampled at low-to-high transitions of TxCLK.
TxADD[4:0]	Transmit address. Address bus from the master to the ATM controller used to select the appropriate M-PHY device.
RxDATA[7:0]	Receive data bus. Carries receive data from the master to the ATM controller. RxDATA[7] is the msb, RxDATA[0] is the lsb.
RxSOC	Receive start of cell. Asserted by the master device whenever the first byte of a cell is being received on the RxDATA lines.
RxENB	Receive enable. Asserted by the master device to signal the slave to sample the RxDATA and RxSOC signals.
RxCLAV	Receive cell available. Asserted by the ATM controller to indicate it can receive a complete cell.
RxPRTY	Receive parity. Asserted by the PHY device. It is an odd parity bit over the RxDATA[7:0]. If there is a RxPRTY error and the receive parity check FPSMR[RxP] is cleared, the cell is discarded. See Section 41.13.3, “FCC Protocol-Specific Mode Register (FPSMR),” and Section 41.10.7, “UNI Statistics Table.”
RxCLK	Receive clock. Provides the synchronization reference for the RxDATA, RxSOC, $\overline{\text{RxENB}}$, RxCLAV, and RxPRTY signals. All the above signals are sampled at low-to-high transitions of RxCLK.
RxADD[4:0]	Receive address. Address bus from master to the ATM controller device used to select the appropriate M-PHY device.

41.12.3.1 UTOPIA Slave Multiple PHY Operation

The user should write the ATM controller PHY address in FPSMR[PHY ID].

41.12.3.2 UTOPIA Clocking Modes

The UTOPIA clock can be generated internally or externally. If the UTOPIA clock is to be generated internally, the user should assign one of the baud-rate generators to supply the UTOPIA clock. See Chapter 24, “CPM Multiplexing.”

41.12.3.3 UTOPIA Loopback Modes

The UTOPIA interface supports loopback mode. In this mode, the Rx and Tx UTOPIA signals are shorted internally. Output pins are driven; input pins are ignored.

Note that in loopback mode, the transmitter and receiver must operate in complementary modes. For example, if the transmitter is master, the receiver must be a slave (FPSMR[TUMS] = 0, FPSMR[RUMS] = 1).

Modes are selected through GFMR[DIAG], as shown in Table 41-45.

Table 41-45. UTOPIA Loopback Modes

DIAG	Description
00	Normal mode
01	Loopback. UTOPIA Rx and Tx signals are shorted internally. Output pins are driven, input pins are ignored.
1x	Reserved

41.13 ATM Registers

The following sections describe the configuration of the registers in ATM mode.

41.13.1 General FCC Mode Register (GFMR)

The GFMR mode field should be programmed for ATM mode. To enable transmit and receive functions, ENT and ENR must be set as the last step in the initialization process. Full GFMR details are given in Section 37.2, “General FCC Mode Registers (GFMRx).”

41.13.2 General FCC Expansion Mode Register (GFEMR*x*)

The general FCC expansion mode registers (GFEMR*x*) define the expansion modes. They should be programmed according to the protocol used.

	0	1	2	3 … 7
Field	TIREM	LPB	CLK	—
Reset	0000_0000			
R/W	R/W			
Offset	0x9_1390 (GFEMR1), 0x9_13B0(GFEMR2)			

Figure 41-60. General FCC Expansion Mode Register (GFEMR)

Table 41-46 describes GFEMR*x* fields.

Table 41-46. GFEMR*x* Field Descriptions

Bits	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode: Internal rate for PHYs[0–3] is controlled only by FTIRR[0–3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode: PHYs[0–31] are controlled by FTIRR[0–3], FIRPER, FIRSR_HI and FIRSR_LO. Underrun status for PHYs[0–31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode mixing of internal rate and external rate is not enabled.
1	LPB	RMII Loopback diagnostic mode (Ethernet mode): 0 Normal mode 1 Loopback mode

Table 41-46. GFEMR*x* Field Descriptions (continued)

Bits	Name	Description
2	CLK	RMII reference clock rate for 50 MHz input clock from external oscillator (Ethernet mode): 0 50 MHz (for Fast Ethernet) 1 5 MHz (for 10BaseT)
3–7	—	Reserved, should be cleared.

41.13.3 FCC Protocol-Specific Mode Register (FPSMR)

The FCC protocol-specific mode register (FPSMR), shown in Figure 41-61, controls various protocol-specific FCC functions. The user should initialize the FPSMR. Erratic behavior may result if there is an attempt to write to the FPSMR while the transmitter and receiver are enabled.

	0–3	4–7	8	9	10	11–15
Field	TEHS	REHS	ICD	TUMS	RUMS	LAST PHY/PHY ID
Reset	0000_0000_0000_0000					
R/W	R/W					
Offset	0x0x9_1304 (FPSMR1), 0x0x9_1324 (FPSMR2)					

	16 17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	TPRI	TUDC	RUDC	RXP	TUMP	—	TSIZE	RSIZE	UPRM	UPLM	RUMP	HECI	HECC	COS
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_1306 (FPSMR1), 0x9_1326 (FPSMR2)														

Figure 41-61. FCC ATM Mode Register (FPSMR)

Table 41-47 describes FPSMR fields.

Table 41-47. FCC ATM Mode Register (FPSMR)

Bits	Name	Description
0–3	TEHS	**Note:** Transmit extra header size. Used only in user-defined cell mode to hold the Tx user-defined cells' extra header size. Values between 0–11 are valid. TEHS = 0 generates 1 byte of extra header; TEHS = 11 generates 12 bytes of extra header.
4–7	REHS	Receive extra header size. Used only in user-defined cell mode to hold the Rx user-defined cells' extra header size. Values between 0–11 are valid. For REHS = 0, the receiver expects 1 byte of extra header; for REHS = 11, it expects 12 bytes of extra header.
8	ICD	Idle cells discard 0 Discard idle cells (GFC, VPI, VCI, PTI =0) 1 Do not discard idle cells
9	TUMS	Transmit UTOPIA master/slave mode 0 Transmit UTOPIA master mode is selected 1 Transmit UTOPIA slave mode is selected

Table 41-47. FCC ATM Mode Register (FPSMR) (continued)

Bits	Name	Description
10	RUMS	Receive UTOPIA master/slave mode 0 Receive UTOPIA master mode is selected 1 Receive UTOPIA slave mode is selected
11–15	LAST PHY/ PHY ID	Last PHY. (Multiple PHY master mode only.) The UTOPIA interface polls all PHYs starting from PHY address 0 and ending with the PHY address specified in LAST PHY. (The number of active PHYs are LAST PHY+1). LAST PHY should be specified in both single-Clav and direct-polling modes. PHY ID. (Multiple PHY slave mode only.) Determines the PHY address of the ATM controller when configured as a slave in a multiple PHY ATM port. **Note:**
16–17	—	Reserved, should be cleared.
18	TPRI	Transmitter priority. Used to adjust the default priority of the FCC transmitter. It is strongly recommended to set TPRI when in multi-PHY mode; for other modes, it should remain cleared. 0 Default operation 1 Prevents elevation to emergency mode Refer to Table 21-5.
19	TUDC	Transmit user-defined cells 0 Regular 53-byte cells 1 User-defined cells
20	RUDC	Receive user-defined cells 0 Regular 53-byte cells 1 User-defined cells
21	RxP	Receive parity check. 0 Check Rx parity line 1 Do not check Rx parity line
22	TUMP	Transmit UTOPIA multiple PHY mode 0 Transmit UTOPIA single PHY mode is selected 1 Transmit UTOPIA multiple PHY mode is selected
23	—	Reserved, should be cleared.
24	TSIZE	Transmit UTOPIA data bus size 0 UTOPIA 8-bit data bus size 1 Reserved
25	RSIZE	Receive UTOPIA data bus size 0 UTOPIA 8-bit data bus size 1 Reserved
26	UPRM	UTOPIA priority mode. 0 Round robin. Polling is done from PHY zero to the PHY specified in LAST PHY. When a PHY is selected, the UTOPIA interface continues to poll the next PHY in order. 1 Fixed priority. Polling is done from PHY zero to the PHY specified in LAST PHY. When a PHY is selected, the UTOPIA interface continues to poll from PHY zero.
27	UPLM	UTOPIA polling mode. 0 Single Clav polling. Polling is done using Add[4:0] and Clav. Selection is done using Add[4:0]. Up to 31 PHYs can be polled. 1 Direct polling. Polling is done using Clav[3:0]. Selection is done using Add[1:0]. Up to 4 PHYs can be polled.

Table 41-47. FCC ATM Mode Register (FPSMR) (continued)

Bits	Name	Description
28	RUMP	Receive UTOPIA multiple PHY mode. 0 Receive UTOPIA single PHY mode is selected 1 Receive UTOPIA multiple PHY mode is selected
29	HECI	HEC included. Used in UDC mode only. 0 HEC octet is not included when UDC mode is enabled. 1 HEC octet is included when UDC mode is enabled.
30	HECC	Receive HEC check 0 Do not check Rx HEC 1 Check Rx HEC. HEC errors are reported in UTOPIAE counter (see Section 41.10.7, "UNI Statistics Table").
31	COS	Coset mode enable 0 Check Rx HEC with no COSET 1 Check Rx HEC with COSET mode enabled

41.13.4 ATM Event Register (FCCE)/Mask Register (FCCM)

The FCCE register is the ATM controller event register when the FCC operates in ATM mode. When it recognizes an event, the ATM controller sets the corresponding FCCE bit. Interrupts generated by this register can be masked in FCCM. FCCE is memory-mapped and can be read at any time. Bits are cleared by writing ones to them; writing zeros has no effect. Unmasked bits must be cleared before the CP clears the internal interrupt request.

FCCM is the ATM controller mask register. The FCCM has the same bit format as FCCE. Setting an FCCM bit enables and clearing a bit masks the corresponding interrupt in the FCCE.

	0–4	5	6	7	8	9	10	11	12	13	14	15
Field	—	TIRU	GRLI	GBPB	GINT3	GINT2	GINT1	GINT0	INTO3	INTO2	INTO1	INTO0
Reset	0000_0000_0000_0000											
R/W	R/W											
Offset	0x0x9_1310 (FCCE1), 0x0x9_1330 (FCCE2), 0x0x9_1314 (FCCM1), 0x0x9_1334 (FCCM2)											

	16–31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Offset	0x9_1312 (FCCE1), 0x9_1332 (FCCE2), 0x9_1316 (FCCM1), 0x9_1336 (FCCM2)

Figure 41-62. ATM Event Register (FCCE)/FCC Mask Register (FCCM)

Table 41-48 describes FCCE fields.

Table 41-48. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	TIRU	Transmit internal rate underrun. A cumulative lag of seven cells has formed between the programmable rate and the actual rate for a specific PHY. A transmit internal rate counter expired and a cell was not sent, either because of slow CPM performance or slow PHY performance. TIRU may be set only when using transmit internal rate mode; see Section 41.13.5, “FCC Transmit Internal Rate Registers (FTIRRx).”
6	GRLI	Global red-line interrupt. GRLI is set when a free buffer pool’s RLI flag is set. The RLI flag is also set in the free buffer pool’s parameter table.
7	GBPB	Global buffer pool busy interrupt. GBPB is set when a free buffer pool’s BUSY flag is set. The BUSY flag is also set in the free buffer pool’s parameter table.
8–11	GINT*x*	Global interrupt. Set when the number of events sent to the corresponding interrupt queue reaches the corresponding event threshold. See Section 41.11, “ATM Exceptions.”
12–15	INTO*x*	Interrupt queue overflow. Set when an overflow condition occurs in the corresponding interrupt queue. This occurs when the CP attempts to overwrite a valid interrupt entry. See Section 41.11.1, “Interrupt Queues.”
16–31	—	Reserved, should be cleared.

41.13.5 FCC Transmit Internal Rate Registers (FTIRR*x*)

The first four PHY devices (address 00–03) on FCC1 and FCC2 have their own transmit internal rate registers (FTIRR*x*_PHY0–FTIRR*x*_PHY3) for use in transmit internal rate mode. In this mode, the total transmission rate is determined by FCC internal rate timers. As a master, the controller only polls the PHY’s Clav status at the rate determined by the internal rate. As a slave, the controller attempts to insert cells into the FIFO at the internal rate. The controller can handle a lag of up to seven cells per PHY between the programmable and actual bus rate. When the cell count mismatch reaches seven, TIRU event is reported, see Section 41.13.4, “ATM Event Register (FCCE)/Mask Register (FCCM).” Note that a mismatch occurs if the PHY rate or the CPM performance are lower then the internal rate. FTIRR*x*, shown in Figure 41-63, includes the initial value of the internal rate timer. The source clock of the internal rate timers is supplied by one of four baud-rate generators selected in CMXUAR; see Section 24.4.1, “CMX UTOPIA Address Register (CMXUAR).” Note that in slave mode, FTIRR*x*_PHY0 is used regardless of the slave PHY address.

	0	1	2	3	4	5	6	7
Field	TRM	Initial Value						
Reset	0000_0000							
R/W	R/W							
Offset	FCC1: 0x0x9_131C (FTIRR1_PHY0), 0x0x9_131D (FTIRR1_PHY1), 0x0x9_131E (FTIRR1_PHY2), 0x0x9_131F (FTIRR1_PHY3) FCC2: 0x0x9_133C (FTIRR2_PHY0), 0x0x9_133D (FTIRR2_PHY1), 0x0x9_133E (FTIRR2_PHY2), 0x0x9_133F (FTIRR2_PHY3)							

Figure 41-63. FCC Transmit Internal Rate Registers (FTIRR*x*)

Table 41-49 describes FTIRR*x* fields.

Table 41-49. FTIRR*x* Field Descriptions

Bits	Name	Description
0	TRM	Transmit mode. 0 External rate mode. 1 Internal rate mode.
1–7	Initial Value	The initial value of the internal rate timer. A value of 0x7F produces the minimum clock rate (BRG CLK divided by 128); 0x00 produces the maximum clock rate (BRG CLK divided by 1).

Figure 41-64 shows how transmit clocks are determined.

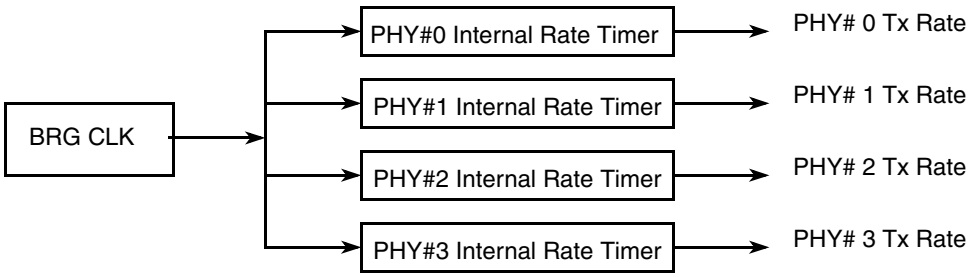


Figure 41-64. FCC Transmit Internal Rate Clocking

Example:

Suppose the MPC8555E is connected to four 155-Mbps PHY devices and the maximum transmission rate is 155 Mbps for the first PHY and 10 Mbps for the rest of the PHYs. The BRG CLK should be set according to the highest rate. If the system clock is 133 MHz, the BRG should be programmed to divide the system clock by 362 to generate cell transmit requests every 362 system clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{155.52\text{Mbps}} = 362$$

For the 155 Mbps PHY, the FTIRR divider should be programmed to zero (the BRG CLK is divided by one); for the rest of the 10 Mbps PHYs, the FTIRR divider should be programmed to 14 (the BRG CLK is divided by 15).

See also Section 41.16.1, “Using Transmit Internal Rate Mode.”

41.14 ATM Transmit Command

The CPM command set includes an ATM TRANSMIT that can be sent to the CP command register (CPCR), described in Section 21.3.1, “CP Command Register (CPCR).”

The ATM TRANSMIT command (CPCR[opcode] = 0b1010, CPCR[SBC[code]] = 0b01110, CPCR[SBC[page]] = 0b00100 or 0b00101) turns a passive channel into an active channel by inserting it into the APC scheduling table. Note that an ATM TRANSMIT command should be issued only after the channel’s TCT is completely initialized and the channel has BDs ready to transmit. Note also that CPCR[SBC[code]] = 0b01110 and not FCC1 or FCC2 code.

Before issuing the command, the user should initialize COMM_INFO fields in the parameter RAM as described in Figure 41-65.

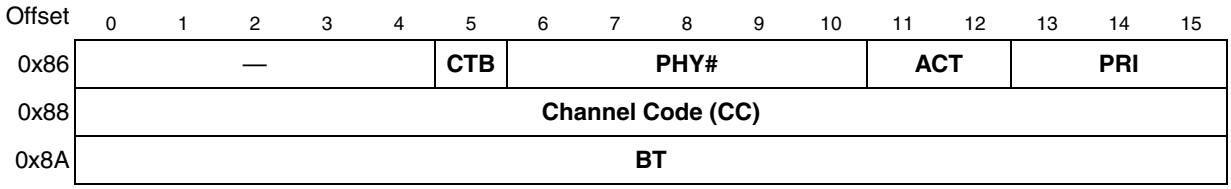


Figure 41-65. COMM_INFO Field

Table 41-50 describes COMM_INFO fields.

Table 41-50. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x86	0–4	—	Reserved, should be cleared.
	5	CTB	Connection tables bus. Used for external channels only 0 External connection tables reside on the system bus. 1 External connection tables reside on the local bus.
	6–10	PHY#	PHY number. In single PHY mode this field should be cleared In multiple PHY mode this field is an index to the APC parameter table associated with this channel.
	11–12	ACT	ATM channel type 00 Other channel 01 VBR channel 1x Reserved
	13–15	PRI	APC priority level. 000 Highest priority (APC_LEVEL1) 111 Lowest priority (APC_LEVEL8).
0x88	0–15	CC	Channel code. The channel code associated with the current channel.
0x8A	0–15	BT	Burst tolerance. For use by VBR channels only (ACT field is 0b01). Specifies the initial burst tolerance (GCRA burst credit) of the current VC.

41.15 Expanded Internal Rate

41.15.1 Transmit External Rate and Internal Rate Modes

The ATM controller supports the following three rate modes:

- External rate mode—The total transmission rate is determined by the PHY transmission rate. The FCC sends cells to keep the PHY FIFOs full; the FCC inserts idle/unassign cells to maintain the transmission rate.
- Internal rate mode—The total transmission rate is determined by the FCC internal rate timers. In this mode, the FCC does not insert idle/unassign cells. The internal rate mechanism is supported for the first four PHY devices (PHY address 0-3). Each PHY has its own FTIRR, described in Section 41.13.5, “FCC Transmit Internal Rate Registers (FTIRRx).” The FTIRR includes the initial value of the internal rate timer. A cell transmit request is sent when an internal rate timer

expires. When using internal rate mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers.

- Internal rate expanded mode—The total transmission rate is determined by the FCC internal rate timers and by the assignment of rate per PHY. In this mode, the FCC does not insert idle/unassign cells. The internal rate expanded mode differs from the internal rate mode in that the internal rate mechanism is extended for 31 PHY devices (PHY addresses 0-30) and there cannot be a mix of external and internal rate PHYs. Expanded internal rate is configured by registers GFEMRx, FIRPERx, FIRSRx_HI, FIRSRx_LO, and by FTIRRx. Another feature of internal rate expanded mode is an indication of transmit underrun error status per PHY. When using internal rate expanded mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers, and any timer can trigger any PHY.

41.15.2 FCC Transmit Internal Rate Mode

In internal rate mode the total transmission rate is the sum of the rates assigned for all PHYs. This register controls how internal rate is configured. In internal rate mode (GFEMR[TIREM] = 0), the internal rate assigned per PHY is configured by registers FTIRR[0–3]. In internal rate expanded mode (GFEMR[TIREM] = 1), registers FTIRR[0–3] control the available rates, but the PHY settings are configured in registers FIRPER, FIRSR_HI and FIRSR_LO. In TIREM = 0 mode internal rate can only be used for PHYs[0–3], whereas in TIREM = 1 mode up to 31 PHYs are supported. If TIREM = 1 mode is selected, the transmit internal rate underrun (TIRU) status per PHY may be read at any time in register FIRER.

41.15.3 FCC Transmit Internal Rate Port Enable Register (FIRPER)

This register enables internal rate transmission for PHYs[0–30]. It is valid only if GFEMR[TIREM] = 1. If a PHY is not enabled in FIRPER, all TxClav indications from that PHY will be masked. The user should configure FIRPER according to the PHY addresses which are being used on the UTOPIA bus and should not enable PHYs with addresses larger then the last PHY address set by FPSMR[Last PHY]. PHYs can be enabled or disabled at any time—for example, if a TIRU event has occurred.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PE0	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9	PE10	PE11	PE12	PE13	PE14	PE15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1380 (FIRPER1), 0x9_13A0 (FIRPER2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	PE16	PE17	PE18	PE19	PE20	PE21	PE22	PE23	PE24	PE25	PE26	PE27	PE28	PE29	PE30	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1382 (FIRPER1), 0x9_13A2 (FIRPER2)															

Figure 41-66. FCC Transmit Internal Rate Port Enable Register (FIRPER)

Table 41-51 describes FIRPER*x* fields.

Table 41-51. FIRPER*x* Field Descriptions (TIREM = 1)

Bit	Name	Description
0–15	PEy	Port enable 0 Transmit internal rate for PHY address y is disabled. TxClav from this PHY is masked. 1 Transmit Internal rate for PHY address y is enabled. The rate assigned for PHY y is selected by register FIRSR_HI (refer to Section 41.15.5, “FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)”).
16–30	PEy	Port enable. 0 Transmit internal rate for PHY address y is disabled. TxClav from this PHY is masked. 1 Transmit Internal rate for PHY address y is enabled. The rate assigned for PHY y is selected by register FIRSR_LO (refer to Section 41.15.5, “FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)”).
31	—	Reserved, should be cleared.

41.15.4 FCC Internal Rate Event Register (FIRER)

Transmit internal rate underrun (TIRU) errors are reported for any PHY that has a transmission deficiency of 8 cells. Under this condition and in internal rate mode only, FCCE[TIRU] is set, and if the corresponding bit in the FCC mask register (FCCM[TIRU]) is set, an interrupt is generated. If TIREM = 1, the TIRU status per PHY can be read at any time in the FCC internal rate event register (FIRER). Once FIRER[TIRUy] error status is set, it can be cleared only by writing 1 to it. To prevent an underrun PHY from continuously reporting errors, it can be disabled by FIRPER. The sequence of disabling a PHY is as follows:

- Disable PHY y by clearing FIRPER[y]
- Clear event FIRER[y] by writing 1 to it
- Clear event FCCE[TIRU] by writing 1 to it

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TIRU 0	TIRU 1	TIRU 2	TIRU 3	TIRU 4	TIRU 5	TIRU 6	TIRU 7	TIRU 8	TIRU 9	TIRU 10	TIRU 11	TIRU 12	TIRU 13	TIRU 14	TIRU 15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1384 (FIRER1), 0x9_13A4 (FIRER2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TIRU 16	TIRU 17	TIRU 18	TIRU 19	TIRU 20	TIRU 21	TIRU 22	TIRU 23	TIRU 24	TIRU 25	TIRU 26	TIRU 27	TIRU 28	TIRU 29	TIRU 30	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_1386 (FIRER1), 0x9_13A6 (FIRER2)															

Figure 41-67. FCC Internal Rate Event Register (FIRER)

Table 41-52 describes FIRER*x* fields.

Table 41-52. FIRER*x* Field Descriptions (TIREM = 1)

Bit	Name	Description
0–30	TIRUy	Transmit internal rate underrun 0 There is no transmission underrun for this PHY. 1 Transmit internal rate underrun or PHY address y has occurred. Bit is cleared by writing 1 to it. Writing 0 has no effect on value.
31	—	Reserved, should be cleared.

41.15.5 FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)

If TIREM = 1, each PHY can be assigned one of four rates, as configured by the four FCC transmit internal rate timers. The FCC internal rate selection registers (FIRSR*x*_HI, FIRSR*x*_LO), shown in Figure 41-68 and Figure 41-69, assign rate group to each of the PHYs.

	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
Field	GS0	GS1	GS2	GS3	GS4	GS5	GS6	GS7
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	0x9_1388 (FIRSR1_HI), 0x9_13A8 (FIRSR2_HI)							

	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
Field	GS8	GS9	GS10	GS11	GS12	GS13	GS14	GS15
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	0x9_138A (FIRSR1_HI), 0x9_13AA (FIRSR2_HI)							

Figure 41-68. FCC Internal Rate Selection Register HI (FIRSR*x*_HI)

Table 41-53 describes FIRSR*x*_HI fields.

Table 41-53. IRSR*x*_HI Field Descriptions (TIREM = 1)

Bit	Name	Description
0–31	GSy	Group select for PHY y 00 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP0. 01 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP1. 10 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP2. 11 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP3.

	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
Field	GS16	GS17	GS18	GS19	GS20	GS21	GS22	GS23
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	0x9_138C (FIRSR1_LO), 0x9_13AC (FIRSR2_LO)							

	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
Field	GS24	GS25	GS26	GS27	GS28	GS29	GS30	—
Reset	0000_0000_0000_0000							
R/W	R/W							
Offset	0x9_138E (FIRSR1_LO), 0x9_13AE (FIRSR2_LO)							

Figure 41-69. FCC Internal Rate Selection Register LO (FIRSR*x*_LO)

Table 41-54 describes FIRSR*x*_LO fields.

Table 41-54. FIRSR*x*_LO Field Descriptions (TIREM = 1)

Bit	Name	Description
0–29	GSy	Group select for PHY y 00 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP0. 01 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP1. 10 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP2. 11 The transmit internal rate for PHY address y is controlled by FTIRR*x*_GRP3.
30–31	—	Reserved, should be cleared.

41.15.6 FCC Transmit Internal Rate Register (FTIRR*x*)

If GFEMR[TIREM] = 0, PHYs at addresses 0–3 have their own FCC transmit internal rate registers (FTIRR*x*_PHY0–FTIRR*x*_PHY3) for use in transmit internal rate mode. If TIREM = 1, FTIRR*x* are used as group timers and PHYs at addresses 0-30 are assigned to a rate group by FIRSR*x*_HI and FIRSR*x*_LO. FTIRR*x*, shown in Figure 41-63, includes the initial value of the internal rate timer. The clock to the internal rate timers is supplied by one of four baud-rate generators selected in CMXUAR; refer to Section 24.4.1, “CMX UTOPIA Address Register (CMXUAR).” Note that in slave mode, FTIRR0 is used regardless of the slave PHY address.

	0	1	7
Field	TRM	Initial Value	
Reset	0000_0000		
R/W	R/W		
Address	GFEMR[TIREM=0] FCC1: 0x9_131C (FTIRR1_PHY0), FCC1: 0x9_131D (FTIRR1_PHY1), FCC1: 0x9_131E (FTIRR1_PHY2), FCC1: 0x9_131F (FTIRR1_PHY3), FCC2: 0x9_133C (FTIRR2_PHY0), FCC2: 0x9_133D (FTIRR2_PHY1), FCC2: 0x9_133E (FTIRR2_PHY2), FCC2: 0x9_133F (FTIRR2_PHY3).	GFEMR[TIREM=1] FCC1: 0x9_131C (FTIRR1_GRP0), FCC1: 0x9_131D (FTIRR1_GRP1), FCC1: 0x9_131E (FTIRR1_GRP2), FCC1: 0x9_131F (FTIRR1_GRP3), FCC2: 0x9_133C (FTIRR2_GRP0), FCC2: 0x9_133D (FTIRR2_GRP1), FCC2: 0x9_133E (FTIRR2_GRP2), FCC2: 0x9_133F (FTIRR2_GRP3).	

Figure 41-70. FCC Transmit Internal Rate Register (FTIRR)

Table 41-49 describes FTIRR*x* fields.

Table 41-55. FTIRR*x* Field Descriptions

Bits	Name	Description
0	TRM (TIREM = 0)	PHY transmit mode 0 External rate mode 1 Internal rate mode
	TRM (TIREM = 1)	Group transmit mode 0 Group rate timer [x] disabled 1 Internal rate timer for Group[x] is enabled and division factor is set by Initial Value field.
1–7	Initial Value	The initial value of the internal rate timer. A value of 0x7F produces the minimum clock rate (BRG CLK divided by 128); 0x00 produces the maximum clock rate (BRG CLK divided by 1).

Figure 41-64 shows how transmit clocks are determined.

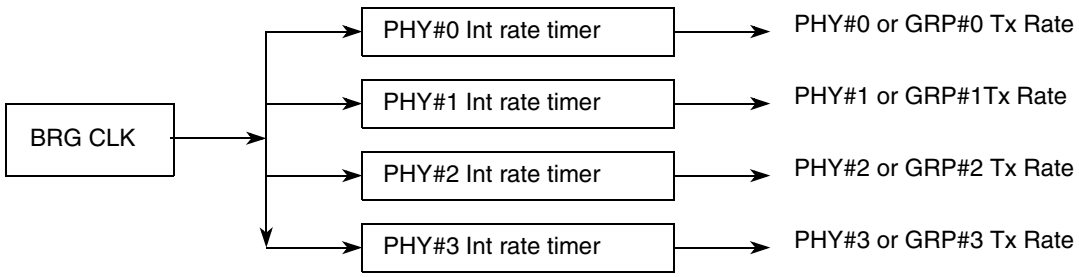


Figure 41-71. FCC Transmit Internal Rate Clocking

41.15.6.1 Example

If the MPC8555E is connected to four 155 Mbps PHY devices and the maximum transmission rate is 155 Mbps for the first PHY and 10 Mbps for the rest of the PHYs, the BRG CLK should be set according to the highest rate. If the system clock is 133 MHz, the BRG should be programmed to divide the system clock by 362 to generate cell transmit requests every 362 system clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{155.52\text{Mbps}} = 362$$

For the 155 Mbps PHY, the FTIRR divider should be programmed to zero (the BRG CLK is divided by one); for the rest of the 10 Mbps PHYs, the FTIRR divider should be programmed to 14 (the BRG CLK is divided by 15).

41.15.7 Internal Rate Programming Model

The programming sequence in TIREM = 0 mode is as follows:

1. Clear GFEMRx[TIREM]
2. Program FTIRRx

The programming sequence in TIREM = 1 mode is as follows:

1. Clear FTIRRx[TRM]
2. Set GFEMRx[TIREM]
3. Program FIRSRx_HI and FIRSRx_LO
4. Program FTIRRx
5. Program FIRPERx

If FTIRRx are set to generate same order of magnitude rates, setting round robin polling mode is more adequate than fixed priority mode. To reduce the risk of transmit underrun if there are a few PHYs with high internal rate and a number of PHYs with a low internal rate, the fast PHYs should be assigned consecutive addresses starting at 0 and fixed priority mode should be chosen.

41.16 Configuring the ATM Controller for Maximum CPM Performance

The following sections recommend ATM controller configurations to maximize CPM performance.

41.16.1 Using Transmit Internal Rate Mode

When the total transmit rate is less than the PHY rate, use the transmit internal rate mode and configure the internal rate clock to the maximum bit rate required. (See 41.2.1.5, “Transmit External Rate and Internal Rate Modes.”) The PHY then automatically fills the unused bandwidth with idle cells, not the ATM controller. If the internal rate mode is not used, CPM performance is consumed generating the idle cell payload and using the scheduling algorithm to fill the unused bandwidth at the higher PHY rate.

For example, suppose a system uses a 155.52-Mbps OC-3 device as PHY0, but the maximum required data rate is only 100 Mbps. In transmit internal rate mode, the user can configure the internal rate mechanism to clock the ATM transmitter at a cell rate of 100 Mbps. If the system clock is 133 MHz, program a BRG to divide the system clock by 563 to generate a transmit cell request every 563 CPM clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{100\text{Mbps}} = 563$$

Set FTIRR*x*_PHY0[TRM] to enable the transmit internal rate mode and clear FTIRR*x*_PHY0[Initial Value] since there is no need to further divide the BRG. See Section 41.13.5, "FCC Transmit Internal Rate Registers (FTIRRx)."

In external rate mode, however, the transmit cell request frequency is determined by the PHY's maximum rate, not by internal FCC counters. If an OC-3 PHY is used with the ATM controller in external rate mode, the requests must be generated every 362 CPM clocks (assuming a 133-MHz CPM clock). If only 100 Mbps is used for real data, 36% of the transmit cell requests consume CPM processing time sending idle cells.

41.16.2 APC Configuration

Maximizing the number of cells per slot (CPS) and minimizing the priority levels defined in the APC data structure improves CPM performance:

- Cells per slot. CPS defines the maximum number of ATM cells allowed to be sent during a time slot. (See Section 41.3.3.1, "Determining the Cells Per Slot (CPS) in a Scheduling Table.") The scheduling algorithm is more efficient sending multiple cells per time slot using the linked-channel field. Therefore, choose the maximum number of cells per slot allowed by the application.
- Priority levels. The user can configure the APC data structure to have from one to eight priority levels. (See Section 41.3.6, "Determining the Priority of an ATM Channel.") For each time slot, the scheduling algorithm scans all priority levels and maintains pointers for each level. Therefore, enable only the minimum number of priority levels required.

41.16.3 Buffer Configuration

Using statically allocated buffers of optimal sizes also improves CPM performance:

- Buffer size. Opening and closing buffer descriptors consumes CPM processing time. Because smaller buffers require more opening and closing of BDs, the optimal buffer size for maximum CPM performance is equal to the packet size (an AAL5 frame, for example).
- Free buffer pool. When the free buffer pool is used, the CPM dynamically allocates buffers and links them to a channel's BD. In static buffer allocation, the core assigns a fixed data buffer to each BD. (See Section 41.10.5.2, "Receive Buffer Operation.") When allowed by the application, use static buffer allocation to increase CPM performance.

Chapter 42
ATM AAL2

The microcode implementation of the ATM adaptation layer type 2 (AAL2) on the MPC8555E is compliant with the ITU-T recommendations I.363.2 and I.366.1. This chapter describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR and should be used as a supplement to Chapter 41, "ATM Controller."

42.1 Introduction

AAL2 enables the multiplexing of voice and data channels over a single ATM VC. The channels consist of packets transported within individual ATM cells (see Figure 42-1). Packet lengths are allowed to vary in order to accommodate bandwidth fluctuations of the individual channels. Each packet has a channel identifier (CID) so that each AAL2 user (channel) is uniquely identified by the triplet VP | VC | CID.

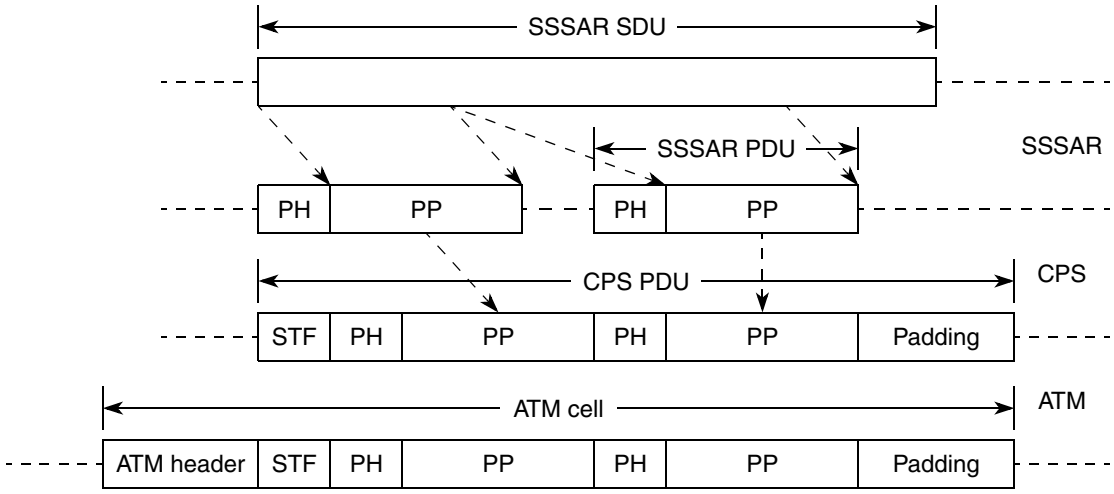
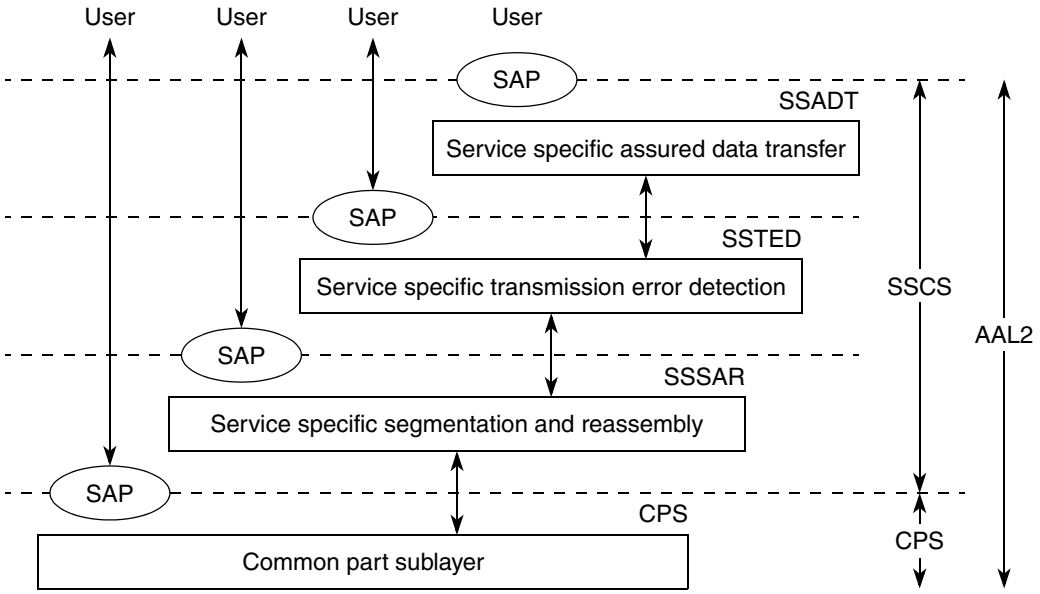


Figure 42-1. AAL2 Data Units

AAL2 is subdivided into two sublayers, as shown in Figure 42-2:

- Common part sublayer (CPS)—In the CPS sublayer, variable length packets coming from multiple users are assembled into CPS-PDUs belonging to a single ATM VC.
- Service-specific convergence sublayer (SSCS)—The SSCS sublayer handles the mapping of user data to the CPS sublayer. The SSCS segments large data frames into smaller CPS packets and also provides different services to the user, such as transmission error detection. The SSCS sublayer is further divided into three service-specific layers:
  - Service-specific segmentation and reassembly sublayer (SSSAR)
  - Service-specific transmission error detection sublayer (SSTED)
  - Service-specific assured data transfer sublayer (SSADT)



(SAP): Service access point

Figure 42-2. AAL2 Sublayer Structure

The AAL2 microcode implements the CPS and SSSAR sublayers. (The SSADT and SSTED sublayers are not implemented.) As shown in Figure 42-2, the user can access the CPS sublayer directly or through the SSSAR sublayer. The SSSAR sublayer is used mainly for transferring large data frames.

The AAL2 microcode also enables switching from one PHY | VP | VC | CID combination to another; an example is shown in Figure 42-3.

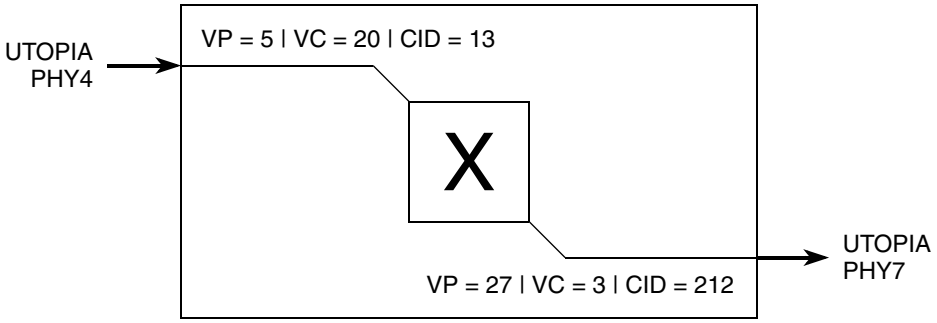


Figure 42-3. AAL2 Switching Example

42.2 Features

The MPC8555E AAL2 features are as follows:

- Fully complies with ITU-T I.363.2 (09/97 and 11/00) and ITU-T I.366.1 (06/98) specifications
- Number of AAL2 external channels supported is subject to internal memory constraints
  - Each external channel requires space for one transmit queue descriptor in internal memory. Typically, up to 1000 external channels can be supported.

- Supports CBR, VBR, and UBR+ traffic types
  - PCR pacing (with optional Timer_CU)
  - VBR pacing (with optional Timer_CU)
  - UBR + pacing (no Timer_CU support)
- Priority mechanism for transmitting per VC. The priority mechanism provides for TX queues having equal or differing priorities. The SSSAR TX queues can be prioritized flexibly among the CPS TX queues.
- Timer_CU support
- NoSTF mode support
- Support for partially filled cells
- User-defined cells (as described in Section 41.7, "User-Defined Cells (UDC)").
- Interrupt indications include the ATM channel number, the CID, and the event type. The events reported are TX buffer not ready, TX buffer transmitted, RX buffer not ready, RX buffer, RX SSSAR frame, and RX AAL2 error events.
- CPS switching
  - Switching from a receive PHY_1 | VP_1 | VC_1 | CID_1 combination to another transmit PHY_2 | VP_2 | VC_2 | CID_2 combination
  - Partial packet discard support
  - For each switched queue, a counter for the total number of packets in the queue is available.
- CPS receiver
  - Segmentation of CPS PDU directly to external memory queues
  - A separate queue for every VP | VC | CID or a common queue for multiple VP | VC | CID combinations
  - Receive one or multiple VP | VC | CIDs directly to a specific TX queue to enable switching
  - Sequence number (SN) protection check for CPS-PDU
  - CRC5 (HEC) check to detect errors in the CPS-PH of the CPS-Packet
  - OSF (offset field) of the STF (start of frame) check (a valid value is less than 48)
  - An SDU length limit parameter (Max_SDU_Deliver_Length) per ATM VC
  - Odd parity check for the STF octet of the CPS-PDU
- CPS transmitter
  - Reassemble CPS PDU directly from external memory
  - Perform CPS-PDU padding as needed
  - Insert sequence number bit of the CPS-PDU
  - Parity bit is calculated to provide odd parity over the STF octet
  - Calculation of CRC-5 on the first 19 bits of the CPS-PH
  - UUI field in the CPS-Packet header is programmed according to the value of the CPS_UUI parameter (per packet)
  - A free running counter (per TX Queue) is decremented for each packet sent.

- SSSAR receiver
  - Reassemble CPS packets from the same CID into an SSSAR SDU
  - A separate queue for every PHY | VP | VC | CID
  - Perform all the above mentioned CPS receiver functions
  - A Ras_Timer mode is provided. When the Ras_Timer expires the buffer is closed with a timer expired error. The next packet received starts a new SSSAR SDU in a new buffer.
  - The SDU length is checked. If the frame exceeds the length limit (SSSAR_Max_SDU_Length), the receiver discards the rest of the packets from the current frame, closes the buffer and reports a Max_SDU violation error in the BD. The next packet received starts a new SSSAR SDU in a new buffer.
  - Partial Packet Discard. If no buffer is available when a packet arrives, the receiver enters a frame hunt state and discards each incoming packet from the current frame.
  - The UUI field is stored for the host into the RxBD after receiving the whole SSSAR frame.
- SSSAR transmitter
  - Segmentation of SSSAR SDUs from SSSAR TX queue into CPS packets
  - An SSSAR TX queue may contain several CIDs
  - Performs all the above mentioned CPS transmitter functions
  - A programmable segment length (Seg_Len) is copied from the SSSAR SDU into the CPS packet payload, except for when at the end of the frame or buffer.
  - UUI mode available. When UUI mode is enabled, the SSSAR UUI is copied from the byte following the last byte of the frame.

42.3 AAL2 Transmitter

The following sections describe the AAL2 transmitter.

42.3.1 Transmitter Overview

A transmitter cycle starts when the APC schedules an ATM channel number for transmission. The TCT is fetched and the AAL type of the channel is checked. For AAL2 cells, the transmitter first handles uncompleted packets from the previous cell of the current CID (partial and split cases) by filling the beginning of the cell with the remainder of the last packet.

Then, the transmitter performs the priority mechanism (see Section 42.3.2, “Transmit Priority Mechanism,”) in order to fill the cell with new packets. The priority mechanism determines the order in which the TX queues are serviced. The transmitter continues to search for ready packets in the TX Queues until either the cell is successfully filled with packets, or no more packets are ready but the cell is not yet completed. In the first case the cell is simply sent. In the latter case, the optional Timer_CU (described in Section 42.3.5.1, “AAL2 Protocol-Specific TCT,”) is examined. If the Timer_CU has expired, the uncompleted cell is padded with zeros and sent; otherwise, the cell is temporarily stored in external memory for the CP to attempt to complete it the next time the channel is scheduled.

The TX queues are the data structures that store the CPS packets and SSSAR frames. Each TX queue can contain different CIDs. Each TX queue is maintained by a Tx queue descriptor (TxQD) that holds the queue pointer and parameters to manage the queue.

When the transmitter fetches a packet out of an SSSAR TX Queue, it usually takes out of the SSSAR buffer a number of octets equal to TxQD[Seg_Len]. (See Section 42.3.5.4, "SSSAR Tx Queue Descriptor.") The channel CID is taken from the BD of the first buffer of the SSSAR frame. (See Section 42.3.5.5, "SSSAR Transmit Buffer Descriptor.") A CPS UUI = 27 is used for all the in-frame packets until the last packet from the SSSAR frame is sent. The last packet can optionally contain a per frame, user-defined UUI. After an SSSAR buffer is completely sent, an optional interrupt event is issued to the host. Also, if an SSSAR TX queue is empty an optional interrupt event is issued to the host.

In case of CPS TX Queue, the transmitter fetches the packet header out of a buffer descriptor and the packet payload out of a CPS buffer. (See Section 42.3.5.3, "CPS Buffer Structure.") The HEC in the packet header is calculated by the CP or taken from the buffer descriptor based on the user configuration. After a CPS packet is sent, an optional interrupt event is issued to the host. Also, if the CPS TX queue is empty an optional interrupt event is issued to the host.

The optional partial filled mode (see Section 42.3.3, "Partial Fill Mode (PFM),") limits the number of data octets per cell. This can be used to ensure that a cell does not contain a split packet or to limit transmission to one packet per TX cell by setting a low partial fill threshold (PFT).

The no-STF (no start of frame) mode (see Section 42.3.4, "No STF Mode,") enables the transmission of cells that do not include the STF byte, thus allowing for 48-byte packets.

42.3.2 Transmit Priority Mechanism

The transmit priority mechanism operates in two modes:

- Round robin (TCT[Fix] = 0)
- Fixed priority (TCT[Fix] = 1)

The following sections describe the priority options.

42.3.2.1 Round Robin Priority

In round robin priority mode, the Tx queues all have equal priority. The transmitter starts with the TxQD pointed to by TCT[FirstQueue], as shown in Figure 42-4. The number of packets that the transmitter services from each queue is determined by the one-packet bit (TCT[OneP]). If TCT[OneP] = 0, the transmitter tries to process as many packets in the queue as needed to fill up the cell. Only when the queue is empty does the transmitter move on to the next queue (assuming the cell is not completed). If TCT[OneP] = 1, the transmitter attempts to take only one packet out of each queue. (Set TCT[OneP] for implementations where each queue contains only one CID.)

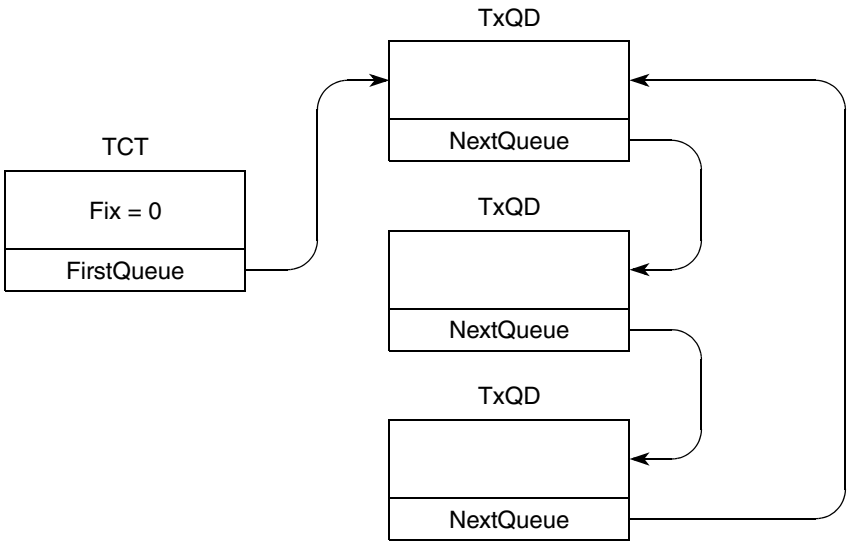


Figure 42-4. Round Robin Priority

The transmitter steps from one TxQD to the next along the queue links. The TCT[MaxStep] parameter limits the number of TX Queues that the transmitter visits during a cell time. If MaxStep is reached before the cell has been completely filled, one of the following events takes place:

- TCT[ET] = 0 (Timer CU disabled). The cell is padded with zeros and sent.
- TCT[ET] = 1 (Timer CU enabled). If the timer has not expired, the cell is not sent. (The transmitter attempts to fill the cell the next time this channel is scheduled.) If the timer has expired, the cell is padded with zeros and sent

After the transmitter sends a cell, it saves the queue link of the last TxQD serviced in TCT[FirstQueue].

42.3.2.2 Fixed Priority

In fixed priority mode (TCT[Fix] = 1), the transmitter, with each new cell, starts with searching the highest priority queue and then moves on to the lower priority queues. The TCT[FirstQueue] points to the highest priority queue, as shown in Figure 42-5, and remains unchanged by the CP.

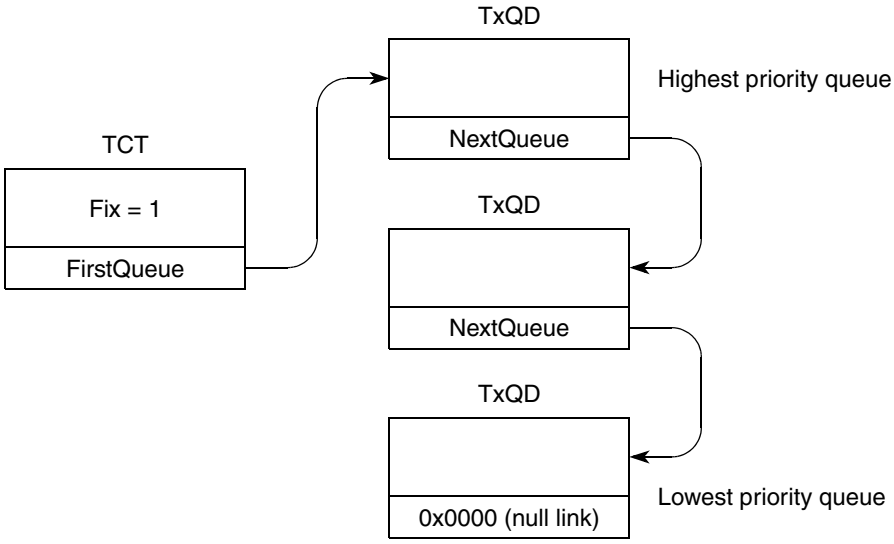


Figure 42-5. Fixed Priority Mode

The TCT[OneP] determines the number of packets that the transmitter attempts to take from each queue (see the explanation in round robin mode).

The NextQueue field of the lowest priority TxQD should be cleared (null link), and TCT[MaxStep] should be programmed to the total number of TX queues in the channel. When the transmitter reaches the null link and the cell is still not complete, the transmitter checks the TIMER CU mode as described above for the round robin mode.

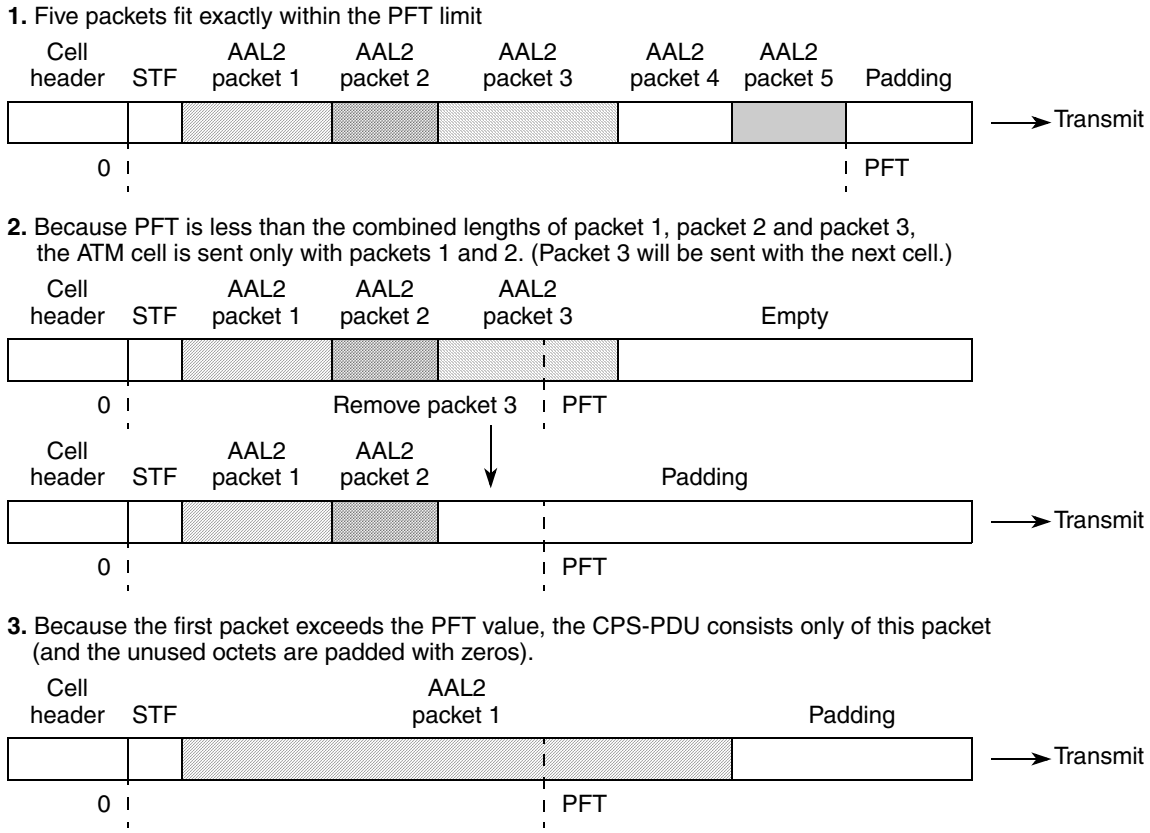
42.3.3 Partial Fill Mode (PFM)

The partial fill mode (TCT[PFM] = 1) allows the user to specify a partial fill threshold (TCT[PFT]), which limits the number of data octets sent with each ATM cell. Partial fill mode assures that packets are not split over two cells, unless the first packet of the cell is greater than 47 bytes.

In partial fill mode, the transmitter starts by filling the ATM cell with the first packet. After the first packet, the CP determines if including the next packet in the cell would exceed the PFT limit. If so, the second packet is not inserted into the current cell, the unused payload is padded with zeros, and the cell is sent. If the second packet does not exceed PFT, the CP inserts it into the CPS-PDU and moves on to the third packet, and so on.

By programming PFT = 1, the partial fill mode can also serve to assure that only one packet is sent per cell.

The following figures provide examples of partial fill mode operation:



42.3.4 No STF Mode

The no-STF (no start of frame) mode enables the transmission of 48-byte packets by not including the STF byte in the CPS PDU. The no-STF mode should always be used with PFT programmed to 47 in order to prevent split and partial packets. For the AAL2 channels that use this mode, packets must not be larger than a maximum packet size of 48.

The user activates this mode per ATM channel by doing the following:

1. Setting TCT[NoSTF] = 1 and TCT[PFM] = 1
2. Establishing a partial fill threshold by programming TCT[PFT] = 47

Figure 42-6 shows a cell using no-STF mode:

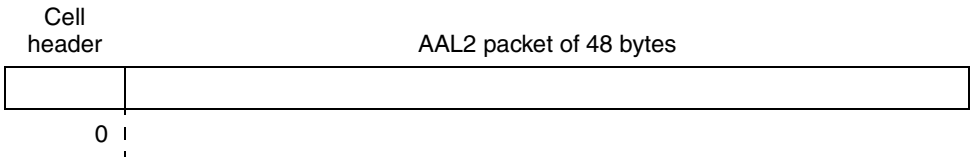


Figure 42-6. Cell in No-STF Mode

42.3.5 AAL2 Tx Data Structures

The following sections describe the transmit connection tables (TCT) and the structures in which CPS packets and SSSAR SDUs are stored in memory.

42.3.5.1 AAL2 Protocol-Specific TCT

The transmit connection table (TCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the TX Queues of the ATM channel are maintained here. Figure 42-7 shows the AAL2-specific TCT.

	0 1	2	3 4	5	6	7	8	9	10 11	12	13	14 15
Offset + 0x00	—	GBL	BO	—	DTB	BIB	AVCF	PFM	ATT	ET	VCON	INTQ
Offset + 0x02	—									NoSTF	AAL	
Offset + 0x04	—											
Offset + 0x06												
Offset + 0x08	—											
Offset + 0x0A	FirstQueue											
Offset + 0x0C	Rate Remainder						PCR Fraction					
Offset + 0x0E	PCR											
Offset + 0x10	Timer_CU_period											
Offset + 0x12	Timer_Period_Shadow											
Offset + 0x14	—											
Offset + 0x16	APC Linked Channel											
Offset + 0x18	ATM Cell Header (VPI,VCI,PTI,CLP)											
Offset + 0x1a												
Offset + 0x1C	—	PMT					MaxStep					
Offset + 0x1E	—	PFT					—			OneP	STPT	Fix PM

Figure 42-7. AAL2 Protocol-Specific Transmit Connection Table (TCT)

NOTE

When the channel is active, the CP fetches the TCT (32 bytes) using burst cycle and writes back only the first (24 bytes).

Table 42-1 describes the AAL2 TCT fields.

Table 42-1. AAL2 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	**GBL**	Global. Setting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	**BO**	Byte ordering. This field is used for data buffers. 00 Reserved 01 Power PC little endian 1x Big endian
	5	—	Reserved, should be cleared during initialization.
	6	**DTB**	Data buffer bus selection. 0 Data buffers reside on the system bus. 1 Reserved.
	7	**BIB**	Bus selection for the BDs, interrupt queues and the free buffer pool. 0 Reside on the system bus. 1 Reserved.
	8	**AVCF**	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more buffers are in the VC's TxBD table, 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears the VCON bit. (Bit 13)
	9	**PFM**	Partial fill mode. See Section 42.3.3, "Partial Fill Mode (PFM)." 0 Partially filled cells are not supported. 1 Partially filled cells are supported.
	10–11	**ATT**	ATM traffic type 00 Peak cell-rate pacing (regular traffic). The host must initialize PCR and PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and Minimum cell rate pacing. The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved.
	12	**ET**	Enable Timer_CU. 0 Timer_CU operation in this channel is disabled. 1 Timer_CU operation in this channel is enabled.
	13	**VCON**	Virtual channel is on Should be set by the host before it issues an ATM TRANSMIT command. When the host sets the STPT (stop transmit) bit, the CP deactivates this channel and clears VCON. The host can issue another ATM TRANSMIT command only when the CP clears VCON.
	14–15	**INTQ**	Points to one of the four interrupt queues available.

Table 42-1. AAL2 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	0–11	—	Reserved, should be cleared during initialization.
	12	**NoSTF**	No STF byte. See Section 42.3.4, "No STF Mode." 0 Normal AAL2 cell structure 1 The cell does not include the STF byte. In this mode each cell starts with a new packet and contains only whole packets (no split or partial).
	13–15	**AAL**	AAL type 000 AAL0—Segmentation with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol 101 AAL1_CES. All others reserved.
0x04	—	—	Reserved, should be cleared during initialization.
0x06	—	—	Reserved, should be cleared during initialization.
0x08	—	—	Reserved, should be cleared during initialization.
0x0A	—	**FirstQueue**	Points to the first queue to be serviced in the transmitter cycle. In round-robin priority mode (TCT[Fix] = 0), this pointer could point to any one of the TxQDs related to this channel. In fixed priority mode (TCT[Fix] = 1), this pointer should point to the highest priority TxQD.
0x0C	0–7	**Rate Remainder**	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared during initialization by the user.
	8–15	**PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	**PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	**Timer_CU_period**	Timer_CU duration in units of APC slots. Assures that CPS-Packet already packed in a cell wait at most the Timer_CU duration. The user defines this field.
0x12	—	**Timer_Period_Shadow**	This field must be initialized to the Timer_CU period in units of APC slots.
0x14	—	—	Reserved, should be cleared during initialization.
0x16	—	**APCLC**	APC linked channel. Used by the CP. Should be cleared during initialization.
0x18	—	**ATMCH**	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	**PMT**	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT×32.
	8–15	**MaxStep**	Holds the number of TX Queues visited for each cell being prepared for transmission. In fixed priority mode (TCT[Fix] = 1), MaxStep should be set to the total number of TX queues in the channel. See Section 42.3.2, "Transmit Priority Mechanism."

Table 42-1. AAL2 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x1E	0–1	—	Reserved, should be cleared during initialization.
	2–7	**PFT**	Partial fill threshold. Used for partially filled cells only; see Section 42.3.3, "Partial Fill Mode (PFM)." Specifies the maximum number of packet bytes allowed in a CPS PDU. The range 1–48 are valid values. If PFT = 48 in partial fill mode, performance is adversely affected. When not in partial fill mode, PFT must be initialized to 47.
	8–11	—	Reserved, should be cleared during initialization.
	12	**OneP**	One packet per queue. See Section 42.3.2, "Transmit Priority Mechanism." 0 The transmitter reads as many packets as possible from each TX queue before moving to the next queue. 1 The transmitter reads only one packet from each TX queue before advancing to the next queue.
	13	**STPT**	Stop transmit. Should be cleared during initialization. When the host sets this bit, the CP removes this channel from the APC and clears TCT[VCON] flag.
	14	**Fix**	Fixed priority. See Section 42.3.2, "Transmit Priority Mechanism." 0 Round robin priority. The TX Queues related to this channel all share the same priority. 1 Fixed priority. The TX queues are ordered in a fixed priority ladder.
	15	**PM**	Performance monitoring 0 No performance monitoring for this VC 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

[1] **Boldfaced** entries must be initialized by the user.

42.3.5.2 CPS Tx Queue Descriptor

Each CPS TxBD table is managed by a CPS Tx queue descriptor (TxQD), as shown in Figure 42-8. The TxQD contains the address of the next BD to be serviced, and other queue-specific parameters. The NextQueue pointer is used to create a linked list of TxQDs, as described in Section 42.3.2, "Transmit Priority Mechanism." The CPS TxQD is located in the dual-port RAM, in a 16-byte aligned address.

	0–7	8	9	10	11	12	13–15
Offset + 0x00	—	**BNM**	**SW**	**HEC**	**CPS**	**TBM**	—
Offset + 0x02	**TxBD Table Offset In** (switched mode only)						
Offset + 0x04	**TxBD Table Base**						
Offset + 0x06							
Offset + 0x08	**TxBD Table Offset Out**						
Offset + 0x0A	**Number of Packets In Queue**						
Offset + 0x0C	**NextQueue**						
Offset + 0x0E	—						

Figure 42-8. CPS Tx Queue Descriptor (TxQD)

Table 42-2 describes the CPS TxQD fields.

.

Table 42-2. CPS TxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	—	Reserved for internal use. (Used to save the BD status of the open BD.)
	8	**BNM**	Buffer not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	**SW**	Switching queue. 0 Normal TX queue 1 This TxQD handles a switching queue. The receiver and transmitter share this queue.
	10	**HEC**	HEC calculation. 0 Transmitter calculates the CPS header HEC. 1 The CPS header HEC is taken as is from the CPS buffer descriptor.
	11	**CPS**	Sublayer type. For a CPS TxQD, this field must be set. 0 SSSAR or SSTED. 1 CPS packet.
	12	**TBM**	Transmit buffer interrupt mask. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue). 1 The transmit buffer event of this queue is enabled.
	13–15	—	Reserved, should be cleared during initialization.
0x02	—	**TxBD Table Offset In**	Used only when this queue is used for switching (SW=1). Should be cleared during initialization.
0x04	—	**TxBD Table Base**	This pointer points to the base address of the BD table.
0x08	—	**TxBD Table Offset Out**	Holds the offset from the TxBD table base to the next BD to be opened by the transmitter. Should be cleared during initialization.

Table 42-2. CPS TxQD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x0A	—	**Number of Packets In Queue**	Counts the number of packets currently in the queue. If this queue is switched, the receiver increments this counter with each new received packet and the transmitter decrements it with each packet sent. For switching, the user should initialize this counter to zero. When this queue is not switched, this counter counts down with every packet sent. (This can have various purposes such as evaluating the packet rate that is transmitted from this queue.).
0x0C	—	**NextQueue**	Points to the next TxQD to be serviced after this one. See Section 42.3.2, "Transmit Priority Mechanism."
0x0E	—	—	Reserved, should be cleared during initialization.

1 **Boldfaced** entries must be initialized by the user.

42.3.5.3 CPS Buffer Structure

The CPS buffer structure consists of a BD table that points to data buffers. The BDs contain, apart from the buffer pointer, also the packet header. The buffers contain the packet payload. See Figure 42-9.

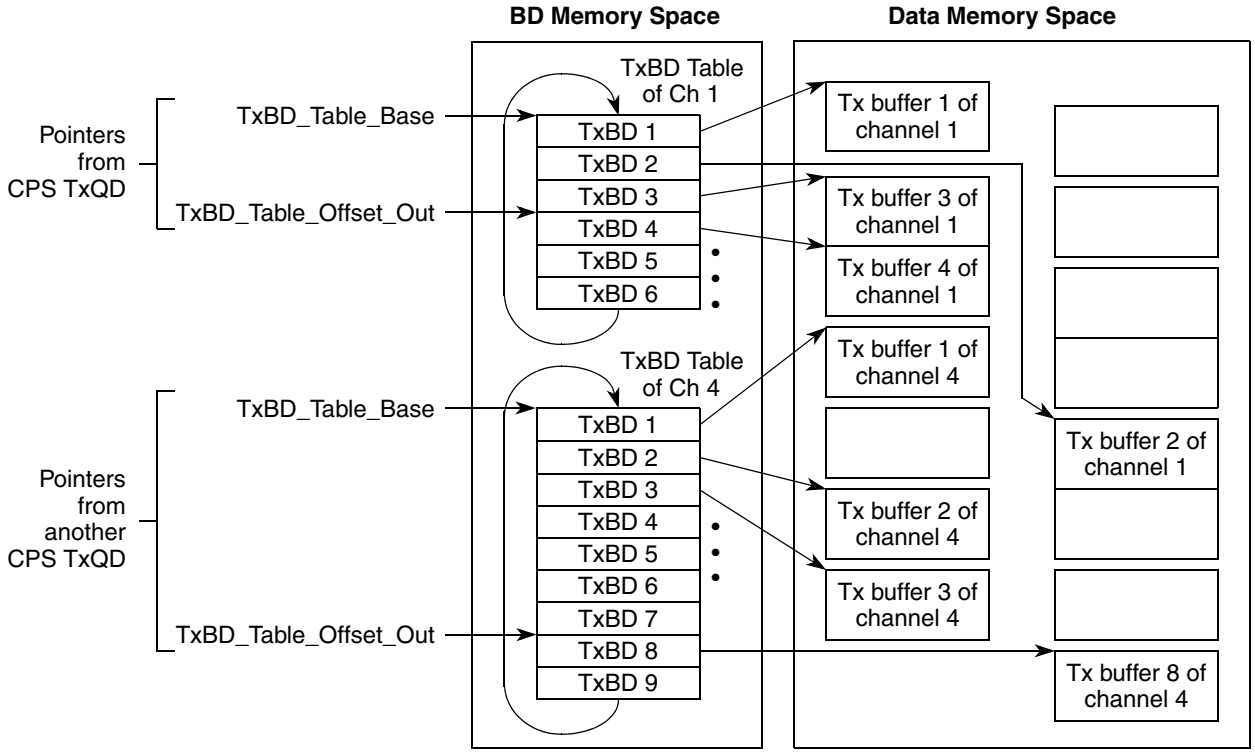


Figure 42-9. Buffer Structure Example for CPS Packets

Figure 42-10 shows a CPS TxBD.

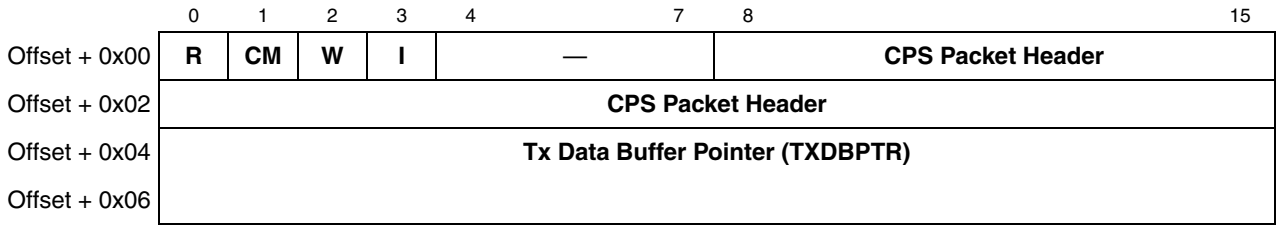


Figure 42-10. CPS TxBD

Table 42-3 describes the CPS TxBD fields.

Table 42-3. CPS TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	**CM**[2]	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4–7	**—**	Reserved, should be cleared during initialization.
	8–15	**CPS Packet Header**	This field contains the beginning (MSB) of the 3-byte packet header. See Figure 42-11 for the CPS packet header format.
0x02	—	**CPS Packet Header**	This field contains the rest of the packet header. If TxQD[HEC] = 0, the HEC part of the packet header is calculated by the CP, and the user may disregard the five least-significant bits of this field. See Figure 42-11 for the CPS packet header format.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This pointer is not modified by the CP.

1 **Boldfaced** entries must be initialized by the user.

2 Setting continuous mode (TxBD[CM] = 1) is not allowed in CID switching mode.

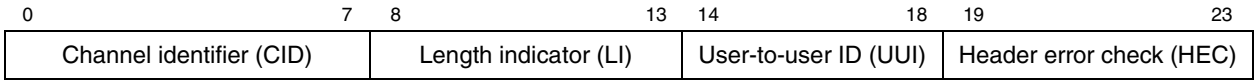


Figure 42-11. CPS Packet Header Format

42.3.5.4 SSSAR Tx Queue Descriptor

A SSSAR TxBD table and its associated buffers are collectively called an SSSAR TX Queue. Each SSSAR TX Queue is managed by an SSSAR TxQD, as shown in Figure 42-12. The TxQD contains the base address of the BD table, the offset of the next BD to be serviced, the data buffer pointer, and other queue-specific parameters. The NextQueue pointer is used to create a linked list of TxQDs, as described in Section 42.3.2, "Transmit Priority Mechanism." The SSSAR TxQD is located in the dual-port RAM in a 32-byte aligned address.

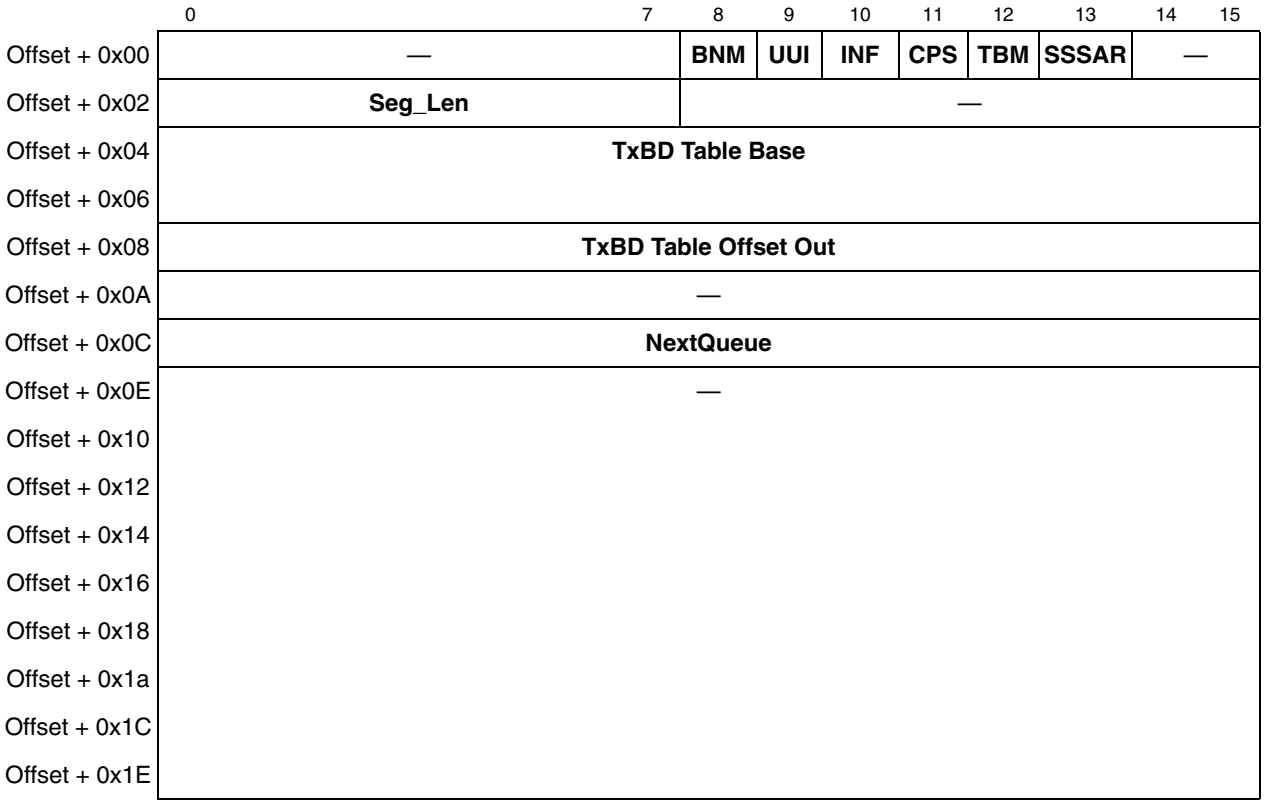


Figure 42-12. SSSAR Tx Queue Descriptor

Table 42-4 describes the SSSAR TxQD fields.

.

Table 42-4. SSSAR TxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	—	Reserved, should be cleared during initialization.
	8	**BNM**	Buffer-not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	**UUI**	UUI insertion mode 0 UUI of last CPS packet is 0. 1 UUI of last CPS packet is taken from the next byte after the end of the buffer.[2]
	10	**INF**	Indicates the current state of the frame. 0 The next packet will be the first of a new frame. 1 Currently in the middle of the frame.
	11	**CPS**	Sublayer type. For an SSSAR TxQD, this field must be cleared. 0 SSSAR or SSTED. 1 CPS packet.
	12	**TBM**	Transmit buffer interrupt mask for TxBD table. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue.) 1 The transmit buffer event of this queue is enabled.
	13	**SSSAR**	SSSAR bit 0 SSTED sublayer 1 SSSAR sublayer
	14–15	—	Reserved, should be cleared during initialization.
0x02	0–7	**Seg_Len**	Specifies the maximum length in bytes of the SSSAR PDU (excluding the packet header). Seg_Len is limited to 45 in NoSTF=1 mode. The CP always attempts to segment the SSSAR SDU according to this length, but not more than it.
	8–15	—	Reserved, should be cleared during initialization.
0x04	—	**TxBD Table Base**	Must be initialized to the first TxBD by the user.
0x08	—	**TxBD Table Offset Out**	Used to calculate the pointer to the next TxBD to be used for transmission. Should be cleared during initialization.
0x0A	—	—	Reserved, should be cleared during initialization.
0x0C	—	**NextQueue**	Points to the next TxQD to be serviced after this one. See Section Section 42.3.2, “Transmit Priority Mechanism.”
0x0E–0x1E	—	—	Reserved, should be cleared during initialization.

1 **Boldfaced** entries must be initialized by the user.

2 The 5-bit UUI field is inserted into the header of the last packet of an SSSAR SDU. The user must append the UUI to the last buffer as an additional byte. This additional byte is then inserted into the UUI field of the last packet header (note that, it is important that this additional byte—the byte after the last byte in the last data buffer of the SSSAR frame—contains the 5 bits of the UUI). The 3 MSB bits of this extra byte should be cleared; refer to Figure 42-11.

42.3.5.5 SSSAR Transmit Buffer Descriptor

The SSSAR buffer structure consists of a BD table that points to data buffers. The buffers may contain SSSAR SDUs belonging to different CIDs. Each buffer may contain a whole SSSAR SDU or part of it. The CPS CID is located in the first BD of the SSSAR SDU. See Figure 42-13.

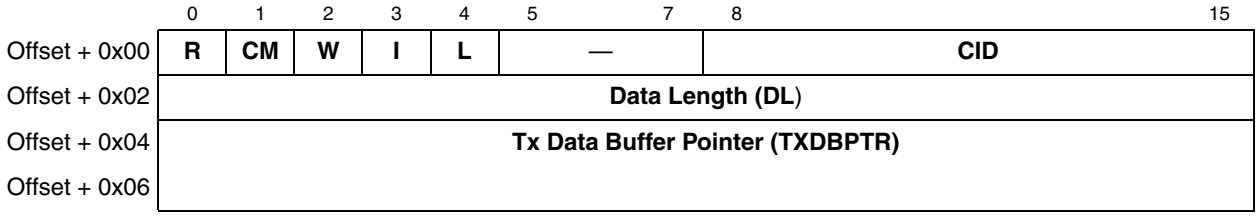


Figure 42-13. SSSAR TxBD

Table 42-5. SSSAR TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	**L**	Last 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–7	—	Reserved, should be cleared during initialization.
	8–15	**CID**	Contains the CID number of the SSSAR SDU pointed by this BD. This field should be written to the first BD of an SSSAR SDU.

Table 42-5. SSSAR TxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	**Data Length**	Contains the length of the buffer associated with this BD. If this is the last buffer (L = 1) and the UUI bit in the SSSAR TxQD is set, the 5-bit UUI field is located at (TXDBPTR+Data Length)[3–7] with bit 3 being the msb, that is, in the byte (right justified) immediately following the last byte of the buffer. For best bandwidth utilization and optimized partitioning of the SDU to packets of exactly Seglen size when an SDU is spread over multiple BD's, the application should set Data Length to be an integer multiple of Seg_Len. (Data Length == n x Seg_Len). For an SDU on a single BD this restriction does NOT apply.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

1 **Boldfaced** entries must be initialized by the user.

42.4 AAL2 Receiver

The following sections describe the AAL2 receiver.

42.4.1 Receiver Overview

The receiver cycle starts after the FCC receives a cell. If the cell header is successfully mapped to an ATM channel number, the corresponding RCT is fetched and the AAL type is read. For AAL2 cells, the receiver begins by checking if the last cell received in this channel (CID) has an uncompleted (split) packet. If so, the receiver first finishes handling this packet.

The receiver then goes through a validation process. The receiver matches the OSF field in the STF with the expected OSF based on the actual split packet (if the first packet is not split, the OSF should be zero). If the two values do not match, an OSF error interrupt is issued and the receiver drops the last packet. Also, if the STF parity check, the SN check or the OSF>47 check results in an error, the receiver issues an interrupt and discards the whole cell. If any of the above errors has occurred and the cell has started with the remainder of an uncompleted packet, the receiver does the following:

- For a CPS sublayer CID, the packet's RxBD[UP] (uncompleted packet) bit is set.
- For an SSSAR sublayer CID, the buffer is closed with RxBD[RxError = US = 10] (uncompleted SDU), and the rest of the frame is dropped.

The receiver now begins the process of extracting new CPS packets out of the cell with another round of error checking. The receiver examines each CPS packet header for the following errors:

- Incorrect packet HEC. The packet and rest of the cell are discarded.
- Packet length (LI+1) is larger than CPS_Max_SDU_Length. The receiver discards the packet and then continues to extract the next packet in the cell. However, if the packet belongs to an SSSAR CID, the receiver closes the SSSAR buffer with RxBD[RxError = OS = 11] (oversized) and discards the rest of the frame.

The receiver issues an interrupt for each of the above errors.When a SSSAR buffer is closed with RxBD[RxError = US or OS], indicating Uncompleted SDU or Oversized, then RxBD[L] is set, and if RxQD[RFM] = 1 then the receiver also issues an RXF interrupt.

Then, if no errors have occurred in the packet header, the packet CID is used to match the PHY | VP | VC | CID with an RxQD; see Section 42.4.2, "Mapping of PHY | VP | VC | CID." The match process yields an RxQD pointer. The RxQD indicates the type of SAR operation to be performed on the PHY | VP | VC | CID. Three SAR operation modes are supported:

- For the CPS sublayer, each packet from the CID is stored, as is, in a one packet buffer.
- For switching, the packet is stored directly into the transmit buffer, and the CID is translated.
- For the SSSAR sublayer, all packets received from the CID are reassembled into an SSSAR SDU similar to the BD and buffer structures used for AAL5 frames.
- For the SSSAR sublayer, last packet UUI indication is stored in the last RxBD of the SDU.

For the SSSAR sublayer, two additional parameters are verified for each new packet received:

- RAS_Timer expiration. The RAS_Timer_Duration defined in the AAL2 parameter RAM limits the time (starting when the first packet is received) allowed to receive a complete SSSAR SDU. If this time limit is exceeded, the receiver closes the current buffer with RxBD[RxError = TE = 01] (Ras_Timer expired) and starts a new SSSAR frame with the next packet. When a buffer is closed with RxBD[RxError = TE = 01], RxBD[L] is not set and the receiver does not issue an RXF interrupt.
- SSSAR_Max_SDU_Length. With each new packet the receiver checks whether the current accumulated length of the SSSAR SDU exceeds the SSSAR_Max_SDU_Length. If so, the receiver closes the current buffer with RxBD[RxError = OS = 11] (oversized), discards the rest of the SSSAR SDU, RxBD[L] is set, and if RxQD[RFM] = 1 issues an RXF interrupt.

NOTE

CIDs that have the same number but that are from different AAL2 connections cannot use the same RxQD, unless they never have split packets.

42.4.2 Mapping of PHY | VP | VC | CID

The AAL2 mapping mechanism translates a PHY | VP | VC | CID combination into an RxQD. An RxQD can be unique per PHY | VP | VC | CID.

The mapping mechanism, shown in Figure 42-14, can be broken down as follows:

- Each ATM channel number (RCT) has its own CID mapping table. The mapping table can be placed in internal or external memory (according to RCT[MAP]) and is pointed to by RCT[CID Mapping Table Base]. The CID of the received packet is used as an index into the mapping table.
- Each entry in the mapping table contains a 2-byte RxQD offset. This offset, multiplied by 4, is the offset to an RxQD in either the internal or external RxQD table.
- The two RxQD tables serve all the ATM channel numbers of an FCC. (RxQD_Base_Int and RxQD_Base_Ext are defined in the FCC parameter RAM.)
- RxQD offsets from 8 through 511 point into the internal RxQD table located in dual-port RAM at RxQD_Base_Int. Note that the first 32 bytes of the internal RxQD table are reserved (so offsets 0–7 are reserved).

- RxQD offsets greater than 511 point into the external RxQD table located at RxQD_Base_Ext + (512 × 4).
- Because the three types of RxQDs are different sizes, some offset numbers may not be used.

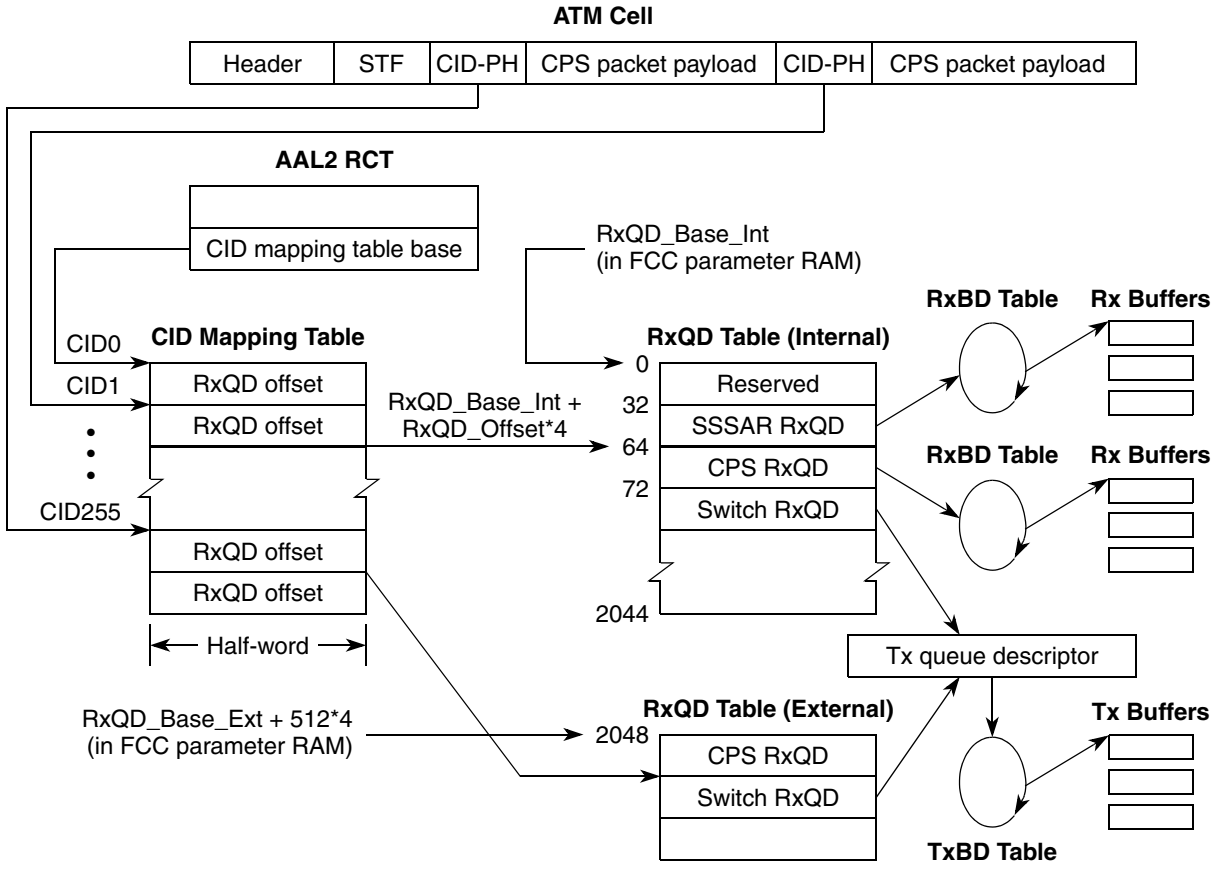


Figure 42-14. CID Mapping Process

42.4.3 AAL2 Switching

Switching is performed by pointing an RX CID at a switch RxQD (see Figure 42-15). The switch RxQD is unique for each Rx CID. The descriptor holds a translation CID number and a pointer to a CPS TxQD into which this packet is saved and later sent by the transmitter. (The TxQD pointer is responsible for the actual PHY | VP | VC switching.) The TxQD pointed to by the switch RxQD(s) should have TxQD[SW] set and should not be modified by the host when the channel is active. The transmit scheduling of the packet is done by the APC according to the programmed bit rate of the ATM channel that holds the switched queue.

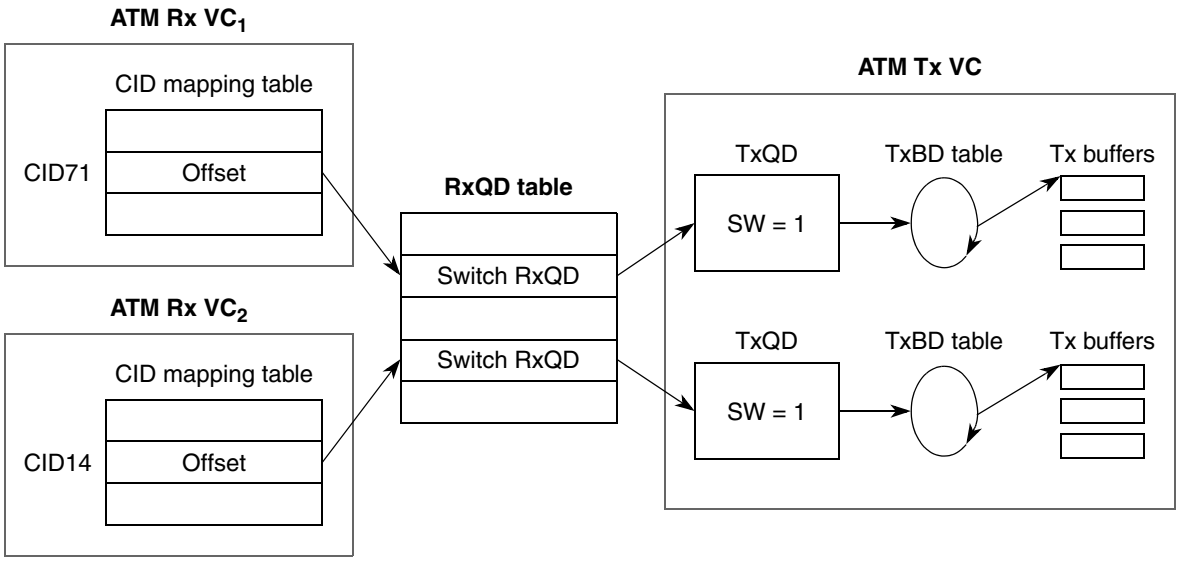


Figure 42-15. AAL2 Switching

A partial packet discard mode is provided for the AAL2 switched channels that perform end-to-end SSSAR. When this mode is enabled (switch RxQD[PPD] = 1), if no buffer is available to receive a packet in the middle of a frame, the subsequent middle packets of the SSSAR SDU are discarded. When the last packet of the SSSAR SDU arrives, the receiver attempts to re-open a buffer.

A number-of-packets-in-queue counter is available in the TxQD. The CP increments the counter for each packet received and decrements it for each packet sent. The host can poll the counter periodically to verify that the switching queues are not over-loaded.

On any open BD that is partially filled, the receiver sets the UP (Un-complete packet) bit. When the packet is fully received during a normal error-free operation, the UP mark is removed, the Empty bit is set, and operation continues. However, if an error is detected by the receiver, the Empty bit is set and the UP bit remains set. In such a case, the transmitter skips this BD and proceeds to the next one. If for any reason the receiver that was in the middle of the BD stopped receiving traffic, the UP bit remains set and the Empty bit is cleared. Another receiver using the same BD ring monitors the UP bit in addition to the Empty bit. If the UP bit is set, the other receiver does not proceed with the reception and gives a Busy interrupt. See Figure 42-18.

The receiver that is in a ‘stuck’ state marks the BD with the receiver channel code that received the partial packet so that the host intervention is easier if needed. See Figure 42-20.

If the TBNR Time Out CNT mechanism is used, the transmitter advances after it tries to transmit the same BD with UP set after a given amount of attempts. The BD will be freed up for use. Refer to the TBNR Time Out CNT description in Table 42-6.

42.4.4 AAL2 Rx Data Structures

The following sections describe the receive connection tables and the structures in which CPS packets and SSSAR SDUs are stored in memory.

42.4.4.1 AAL2 Protocol-Specific RCT

The receive connection table (RCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the RX Queues of the ATM channel are maintained here. The RCT also contains the pointer to the CID mapping table for the ATM VC. Figure 42-16 shows the AAL2-specific RCT.

	0 1	2	3 4	5	6	7	8 9	10	11	12	13	14 15
Offset + 0x00	—	GBL	BO	—	DTB	BIB	—	SEGF	ENDF	—	MAP	INTQ
Offset + 0x02	—									NoSTF	AAL	
Offset + 0x04	—											
Offset + 0x06												
Offset + 0x08												
Offset + 0x0A												
Offset + 0x0c												
Offset + 0x0e												
Offset + 0x10												
Offset + 0x12												
Offset + 0x14												
Offset + 0x16												
Offset + 0x18	CID Mapping Table Base											
Offset + 0x1A												
Offset + 0x1C	—	PMT					TBNR Time OUT CNT					
Offset + 0x1E	Max_CPS_SDU_Deliver_Length						—				EM	PM

Figure 42-16. AAL2 Protocol-Specific Receive Connection Table (RCT)

NOTE

For an active channel, the CP uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes.

Table 42-6 describes the AAL2 RCT fields.

Table 42-6. AAL2 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name [1]	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	**GBL**	Global. Setting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	**BO**	Byte ordering—used for data buffers. 00 Reserved 01 Munged little endian 1x Big endian
	5	—	Reserved, should be cleared during initialization.
	6	**DTB**	Data buffer bus selection. 0 Reside on the system bus 1 Reserved.
	7	**BIB**	Bus selection for the BDs, interrupt queues, CID mapping table, RxQDs, and the free buffer pool. 0 Reside on the system bus 1 Reserved.
	8–9	—	Reserved, should be cleared during initialization.
	10	**SEGF**	OAM F5 segment filtering 0 Do not send cells with PTI = 100 to the raw cell queue. 1 Send cells with PTI = 100 to the raw cell queue.
	11	**ENDF**	OAM F5 end-to-end filtering 0 Do not send cells with PTI = 101 to the raw cell queue. 1 Send cells with PTI = 101 to the raw cell queue.
	12	—	Reserved, should be cleared during initialization.
	13	**MAP**	CID mapping table memory location select 0 Resides in the dual-port RAM. 1 Resides in external memory.
	14–15	**INTQ**	Assigns one of the four available interrupt queues to this ATM channel number.
0x02	0–11	—	Reserved, should be cleared during initialization.
	12	**NoSTF**	No STF byte 0 Normal AAL2 cell structure 1 The cell does not include the STF byte. In this mode each cell starts with a packet and contains only whole packets (no split or part).
	13–15	**AAL**	AAL type 000 AAL0—Reassembly with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol 101 AAL1_CES. All others reserved.

Table 42-6. AAL2 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name [1]	Description
0x04	—	—	Reserved, should be cleared during initialization.
0x06	—		
0x08	—		
0x0A	—		
0x0C	—		
0x0E	—		
0x10	—		
0x12	—		
0x14	—		
0x16	—		
0x18	—	**CID Mapping Table Base**	Points to the base address of the CID mapping table (see Figure 42-14). If RCT[MAP] = 0, the pointer contains a dual-port RAM address and only the 16 lsb (at 0x1A and 0x1B) are relevant. If MAP = 1, the pointer is a full 32-bit address in the memory space.
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	**PMT**	Performance monitoring table. Assigns one of the available 64 performance monitoring tables to this VC. The table's starting address is PMT_BASE+PMT×32.
	8–15	**TBNR Time Out CNT**	The TBNR Time-Out CNT is a parameter that describes the amount of attempts the transmitter tries to transmit a packet on a BD ring which is current marked as partially filled, that is, waiting for a receiver to finish reception of a packet. This value will be used internally by the transmitter that is the destination for this packet and decremented by the CPM on each attempt. Upon reaching the value 1, the transmitter will act as if the receiver is stuck in error condition and proceed to the next BD in the BD ring. This parameter is valid in switch mode only and should be programmed to a higher value than the ratio between the transmitter rate and the lowest receiver rate in the BD ring. The 8-bit value is scaled by 4 (setting TBNR Time-Out CNT =1 yields a value of 4 internally) so that the max number is 1K. Clearing this field will disable this feature completely
0x1E	0–7	**Max_CPS_ SDU_Deliver _Length**	Indicates the maximum size CPS_SDU in bytes that is allowed to be transported on this channel. This value is compared to the length of each CPS_SDU before it is delivered, as specified in the ITU-T recommendation I.363.2.
	8–13	—	Reserved, should be cleared during initialization.
	14	**EM**	Receive error mask for AAL2 protocol-specific events. Note that buffer-not-ready, Rx buffer and Rx frame events are not affected by this mask. 0 Disable AAL2 receive error events. 1 Enable AAL2 error events.
	15	**PM**	Enable performance monitoring 0 No performance monitoring for this VC. 1 Perform performance monitoring for this VC. Whenever a cell is received by this VC, the associated performance monitoring table is updated.

[1] **Boldfaced** entries must be initialized by the user.

42.4.4.2 CID Mapping Tables and RxQDs

Each PHY | VP | VC | CID combination is assigned an RxQD using a CID mapping table. To multiplex several receive CIDs into a single common queue, map each multiplexed PHY | VP | VC | CID combination to one RxQD.

The ATM channel's RCT contains the base address of the associated CID mapping table. This base address is external (32 bits) when RCT[MAP] = 1; otherwise, the table resides in the dual-port RAM and the base address is two bytes. The CID of the received packet is used as an index into the mapping table. The mapping table entries are 2-byte RxQD offsets. If the CID mapping table is external, it must be on the same bus as the BDs and interrupt queues as specified by RCT[BIB].

There are two RxQDs—one for internal RxQDs and one for external RxQDs. Offsets between 0–511 belong to the 2048-byte internal RxQD table. It is recommended to have as many RxQDs as possible in the internal table. Note that the first 32 bytes of the internal RxQD table are reserved for internal use; that is, RxQD offsets between 0–7 are reserved. The address of an internal RxQD is RxQD_Base_Int + 4 × RxQD_Offset.

Offsets between 512–65535 belong to the external RxQD table. The address of an external RxQD is RxQD_Base_Ext + 4 × RxQD_Offset. The external RxQD table must be on the same bus as the BDs and interrupt queues as specified by RCT[BIB].

Because the three kinds of RxQDs are each a different size (for example, an SSSAR RxQD is 32 bytes and a CPS switch RxQD is only 4 bytes), some of the offset numbers are left unused.

42.4.4.3 CPS Rx Queue Descriptors

Each CPS RxQD, as shown in Figure 42-17, points to an CPS RxBD table.

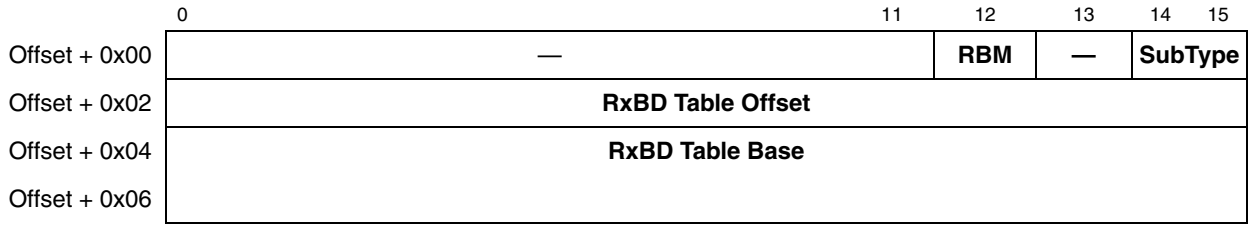


Figure 42-17. CPS Rx Queue Descriptor

Table 42-7 describes the CPS RxQD fields.

Table 42-7. CPS RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–11	—	Reserved, should be cleared during initialization.
	12	**RBM**	Receive buffer mask 0 Disable receive buffer interrupt 1 Enable receive buffer interrupt
	13	—	Reserved, should be cleared during initialization.
	14–15	**SubType**	SubType. Sublayer type, should be 00 (CPS) for this descriptor 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved.
0x02	—	**RxBD Table Offset**	Holds the offset to the next BD to be opened by the receiver. Should be cleared during initialization.
0x04	—	**RxBD Table Base**	Holds the pointer to the first BD in the BD table

1 **Boldfaced** entries must be initialized by the user.

42.4.4.4 CPS Receive Buffer Descriptor (RxBD)

The CPS RxBD structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 42-18. The buffers contain the packet payload.

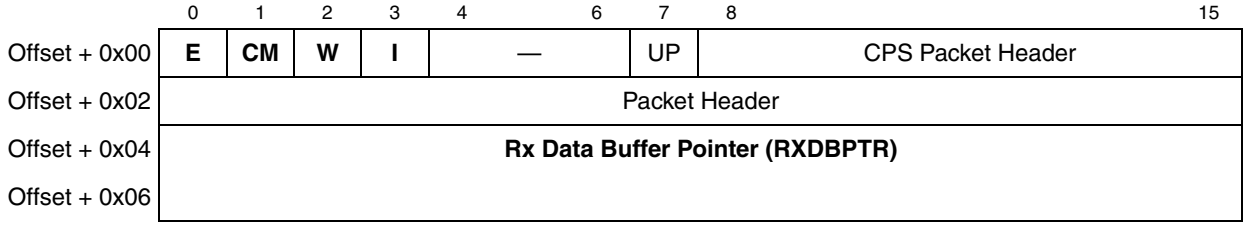


Figure 42-18. CPS Receive Buffer Descriptor

Table 42-8 describes the CPS RxBD fields.

Table 42-8. CPS RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Buffer empty bit 0 The CPS RX buffer is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E remains zero. 1 The CPS RX buffer is empty or reception is in progress. This is controlled by the CP. Once E is set, the core should not access any fields of this buffer.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the CP next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 The CP will not issue an interrupt after this buffer is serviced. 1 The CP will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4–6	—	Reserved, should be cleared during initialization.
	7	UP	Uncompleted packet 0 No error occurred in this packet 1 A receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue.
	8–15	CPS Packet Header	Contains the beginning of the packet header. See Figure 42-11 for the CPS packet header format.
0x02	—	CPS Packet Header	Contains the rest of the packet header. The CP checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 42-11 for the CPS packet header format.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

42.4.4.5 CPS Switch Rx Queue Descriptor

The switch RxQD, shown in Figure 42-19, is used for CIDs that are being switched from one $PHY_1 \mid VP_1 \mid VC_1 \mid CID_1$ to another $PHY_2 \mid VP_2 \mid VC_2 \mid CID_2$. The RxQD contains the pointer to the TxQD that controls the TxBD table through which the packet is transferred.

The switch RxQD also contains the translation CID that is saved with the packet in the transmit buffer. A PPD mode enables the discarding of the rest of an SSSAR frame when a buffer is not available.

Note that the CPS switch RxQD must be unique for every Rx switched CID.

Figure 42-19. CPS Switch Rx Queue Descriptor

Table 42-9 describes the CPS switch RxQD fields.

Table 42-9. CPS Switch RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	**TX CID**	Translation CID. The received CID is saved in a TX Queue with this new CID number.
	8–11	**—**	Reserved, should be cleared during initialization.
	12	**RBM**	Receive buffer mask 0 Disable receive buffer interrupt 1 Enable receive buffer interrupt
	13	**PPD**	Partial packet discard 0 Normal mode 1 When a buffer-not-ready event causes a packet to be discarded, the remainder of the SSSAR SDU is also discarded. This allows for better performance for switched channels that implement SSSAR.
	14–15	**SubType**	Sublayer type. Should be 01 (CPS switched) for this descriptor. 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved
0x02	—	**TxQD Pointer**	Points to the TxQD into which the packets of this CID are stored and later sent.

1 **Boldfaced** entries must be initialized by the user.

42.4.4.6 SWITCH Receive/Transmit Buffer Descriptor (RxBD)

The switch buffer structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 42-20. The buffers contain the packet payload. This BD is common to the receiver and the transmitter.

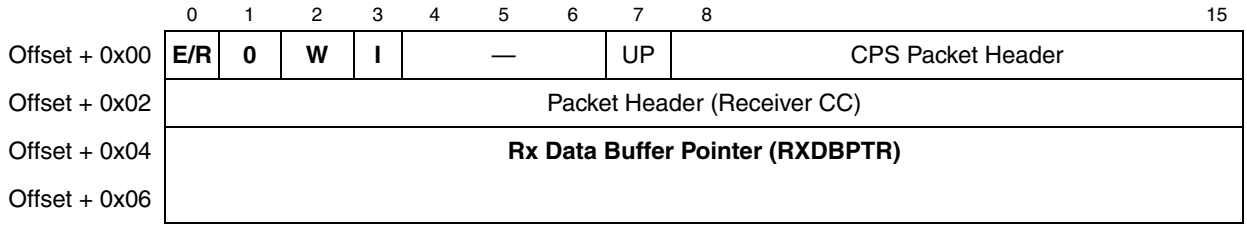


Figure 42-20. Switch Receive/Transmit Buffer Descriptor

Table 42-10 describes the Switch RxBD fields.

Table 42-10. Switch RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E/R**	Buffer ready Must be set to zero.
	1	**0**	Not valid for switching mode, should be cleared to 0 on initialization.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt. 0 The CP will not issue an interrupt after this buffer is serviced. 1 The CP will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4–6	—	Reserved, should be cleared during initialization.
	7	UP	Uncompleted packet. 0 No error occurred in this packet and the complete packet has been received. 1 If R/E = 1 a receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue. The transmitter will skip this BD when in this state and continue to the next BD in the ring. If R/E = 0 a receiver has received the first part of a packet and is waiting for the rest of it to be received on the next ATM cell.
	8–15	CPS Packet Header	Contains the beginning of the packet header. See Figure 42-11 for the CPS packet header format. (see remark in next row)
0x02	—	CPS Packet Header (Receiver CC)	Contains the rest of the packet header. The CP checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 42-11 for the CPS packet header format. In case of a ‘stuck’ receiver in switch mode, where the BD ring in common to Tx and Rx, this field indicates the last Receiver Channel Code number which has been received. The terminology for ‘stuck’ implies a receiver which started receiving a packet and the rest of the packet hasn’t been received.When the receiver is in a ‘stuck’ state the entry: CPS Packet Header is not valid. If the Time-out mechanism is being used this field is being used internally by the CPM.
0x04	—	**RXBDPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

42.4.4.7 SSSAR Rx Queue Descriptor

The SSSAR RxQD, as shown in Figure 42-21, points to the RxBD table and contains other parameters specific to the SSSAR sublayer. This descriptor can belong to only one PHY | VP | VC | CID.

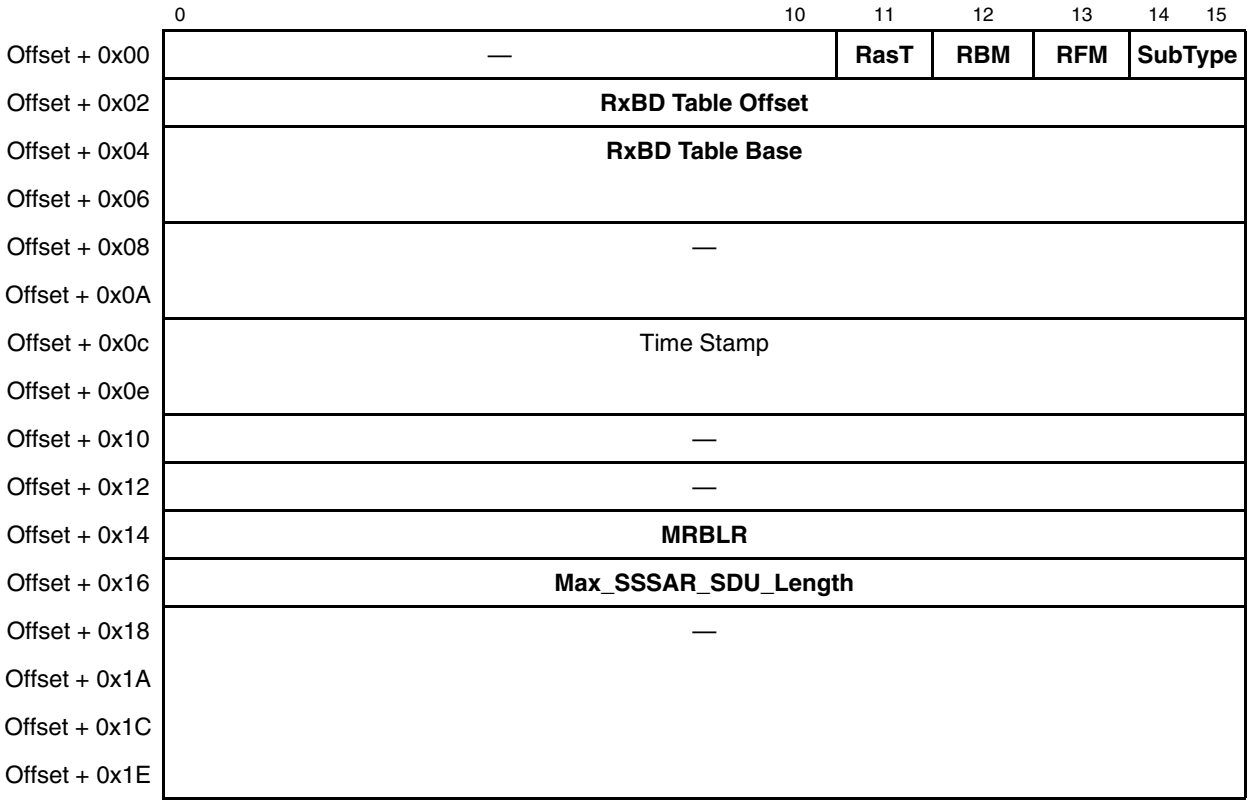


Figure 42-21. SSSAR Rx Queue Descriptor

Table 42-11 describes the SSSAR RxQD fields.

Table 42-11. SSSAR RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–10	—	Reserved, should be cleared during initialization.
	11	**RasT**	Ras timer enable 0 Ras Timer disabled (time stamp field is still valid) 1 Ras Timer enabled. The Ras Timer duration is set by the Ras Timer Duration parameter in the parameter RAM. If the current SSSAR SDU is not completed before the RasTimer expires, the BD is closed showing the Ras_Timer expired (TE) (SSSAR RxBD[RxError] = 01) and the next packet starts a new SDU.
	12	**RBM**	Receive buffer mask 0 Disable receive buffer interrupt 1 Enable receive buffer interrupt
	13	**RFM**	Receive frame mask 0 Disable receive frame interrupt 1 Enable receive frame interrupt
	14–15	**SubType**	Sublayer type. Should be 10 (SSSAR) for this descriptor. 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved

Table 42-11. SSSAR RxQD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	**RxBD Table Offset**	Points to the next BD to be handled by the CP. The user should initialize this pointer to zero.
0x04	—	**RxBD Table Base**	Points to the beginning of the BD table.
0x08	—	—	Reserved, should be cleared during initialization.
0x0A	—	—	Reserved, should be cleared during initialization.
0x0C	—	**Time Stamp**	Used for reassembly timeout of the SSSAR SDU. Whenever the first packet of an SSSAR SDU arrives the timestamp timer is sampled and stored here (regardless of the RasT bit).
0x10	—	—	Reserved, should be cleared during initialization.
0x12	—	—	Reserved, should be cleared during initialization.
0x14	—	**MRBLR**	Maximum receive buffer length. Holds the maximum receive buffer length. The actual buffer size can be less.
0x16	—	**Max_SSSAR_ SDU_ Length**	Holds the maximum SSSAR SDU length. Upon each new packet the accumulated frame size is compared with this value. If the limit is exceeded, the CP discards the rest of the packets of the current frame.
0x18–0 x1E	—	—	Reserved, should be cleared during initialization.

1 **Boldfaced** entries must be initialized by the user.

42.4.4.8 SSSAR Receive Buffer Descriptor

The SSSAR SDU is stored in a BD-buffer structure similar to the structures used for AAL5 frames. The buffer size is determined by SSSAR RxQD[MRBLR]; the actual buffer space used may be smaller. If the received SSSAR SDU is greater than MRBLR, the SDU spans over multiple buffers.

The SSSAR RxBD is shown in Figure 42-22.

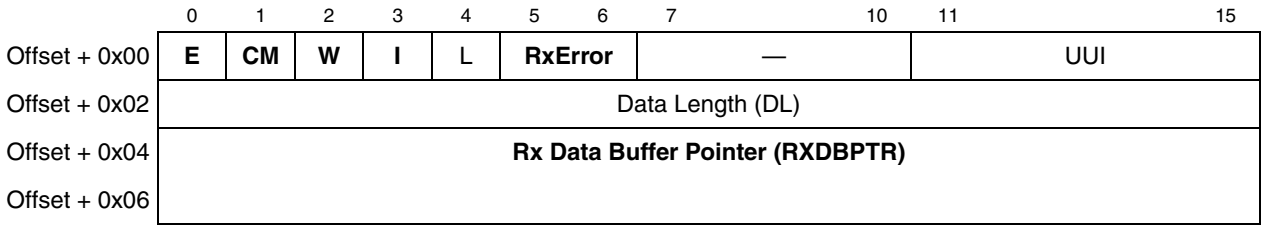


Figure 42-22. SSSAR Receive Buffer Descriptor

Table 42-12 describes the SSSAR RxBD fields.

Table 42-12. SSSAR RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Empty 0 The buffer associated with this RxBD is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The buffer associated with this RxBD is empty or reception is in progress. This RxBD and its receive buffer are controlled by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the CP next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in the table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 An Rx buffer event is sent (provided that RxQD[RBM] is set) to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	L	Last. Set by the CP. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–6	RxError	Rx error occurred 00 No Rx error occurred 01 TE—Ras_Timer expired. The Ras Timer expired before this buffer could be completed. The SSSAR SDU stored in this buffer is not completed.[2] 10 US—Uncompleted SDU. A receive error caused a packet belonging to this SSSAR SDU to be lost. The receiver discarded the rest of this SSSAR SDU. 11 OS—Oversized. The size of the SSSAR SDU has exceeded the Max_SSSAR_SDU_Size parameter. The rest of the SDU was discarded.
	7–10	—	Reserved, should be cleared during initialization.
	11–15	UUI	Contains the UUI of the last packet in the received SDU. Valid only where the L bit is set.

Table 42-12. SSSAR RxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	Data Length	Contains the length of the buffer associated with this BD. If this is the last buffer (L = 1) of the SSSAR SDU, this field contains the total frame length.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

1 **Boldfaced** entries must be initialized by the user.

2 When RAS timer expires the RxBD is closed with RAS timer expired indication, the Last (L) bit is not set, and RXF interrupt is not issued. A new RxBD is opened for the next incoming AAL2 packet and the frame is processed as normal and is treated as a new frame. When the next SSSAR end-of-frame indication is received, the RxBD at that time is closed with an L indication, and if RxQD[RFM] = 1, the receiver issues an RXF interrupt.

42.5 AAL2 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 42-13. The table includes both the fields for general ATM operation and also the fields specific to AAL2 operation.

Note that some of the values must be initialized by the user, but values updated by the CP should not be modified by the user.

Table 42-13. AAL2 Parameter RAM

Offset	Name	Width	Description
0x00–0x3F	—	—	Reserved. Should be cleared during initialization.
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 32 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined.
0x4C	RAS_Timer_Duration	Word	Contains the RAS_Timer duration in microseconds for the SSSAR sublayer. User-defined.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined.

Table 42-13. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. The offset of the UEAD entry in the UDC extra header. Should be an even address. If RCT[BO]=01 UEAD_OFFSET should be in little-endian format. For example if UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be set to 2 (second half word entry in internal RAM).
0x5E	RxQD_Base_Int	Hword	Points to the base address of the internal RxQD table. The pointer should be 32-byte aligned. User-defined.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined.
0x62	APCP_BASE	Hword	APC parameters table base address. User-defined.
0x64	FBT_BASE	Hword	Free buffer pool parameters table base. User-defined.
0x66	INTT_BASE	Hword	Interrupt queue parameters table base. User-defined.
0x68	—	—	Reserved. Should be cleared during initialization.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0–7] hold the 8 most-significant bits of the RX/TxBD table base address. BD_BASE_EXT[8–31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCI_Filtering	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7-15 are received and the associated VCI_Filtering bit = 1 the cell is sent to the raw cell queue. VCI=3 is associated with VCI_Filtering[3], VCI = 15 is associated with VCI_Filtering[15]. VCI_Filtering[0–2, 5] should be zero. See Section 41.10.1.2, “VCI Filtering (VCIF).”
0x84	GMODE	Hword	Global mode. User-defined. See Section 41.10.1.3, “Global Mode Entry (GMODE).”
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 41.14, “ATM Transmit Command.”
0x88		Hword	
0x8A		Hword	
0x8C	—	Word	Reserved. Should be cleared during initialization.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFFFFFF.
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SN protection look up table base address. (AAL1 only.) The 32-byte table resides in dual-port RAM and must be initialized by the user. (See Section 41.10.6, “AAL1 Sequence Number (SN) Protection Table.”)

Table 42-13. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0x9A	—	Hword	Reserved. Should be cleared during initialization.
0x9C	SRTS_BASE	Word	External SRTS logic base address. (AAL1 only.) Should be 16-byte aligned.
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP=1).
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/Unassign cell size. 52 in regular mode. 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A6A6A.
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler (See RTSCR). For time stamp prescaler of 1µs, Trm should be set to 100 ms/1µs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less than TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be 16-byte aligned. User-defined.
0xB4	RxQD_Base_Ext	Word	Points to the base address of the external RxQD table. The actual address of the first RxQD in the table is RxQD_Base_Ext + 512 × 4. User-defined.
0xB8	RX_UDC_Base	Word	Valid only for AAL2 VCs. Points to the base of the RX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: RX_UDC_Base + 16*CH# (where CH# is the ATM channel number)
0xBC	TX_UDC_Base	Word	Valid only for AAL2 VCs. Points to the base of the TX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: TX_UDC_Base + 16*CH# (where CH# is the ATM channel number)
0xC0–0xDF	—	—	Reserved. Should be cleared during initialization.
0xE0	TCELL_TMP_BASE_EXT	Word	Transmit Cell Temporary base address. Points to a total of 64*last_AAL2_Ch# octets reserved in external memory for partially filled cells. Note: TCELL_TMP_BASE_EXT must be on the same bus as the all the AAL2 data buffers required for CPS, SSSAR, and CID switching.
0xE4–0xFB	—	—	Reserved. Should be cleared during initialization.
0xFC	PAD_TMP_BASE	Hword	PAD template base address. Points to an internal memory area that contains the zero cell template. Should be 64-byte aligned. User-defined.
0xFE	—	—	Reserved. Should be cleared during initialization.

42.6 User-Defined Cells in AAL2

The user-defined cell (UDC) mode for ATM as described in Section 41.7, “User-Defined Cells (UDC),” also applies to AAL2 operation. However, for AAL2 operation only, the UDC headers reside in a table in external memory, not in the BDs.

For transmit channels in AAL2 UDC mode, initialize its UDC header entry in the TX UDC header table before activating the channel. The header can be up to 12 bytes. The TX_UDC_Base parameter in the parameter RAM (see Table 42-13), points to the beginning of the TX UDC header table.

The UDC header of a specific AAL2 transmit VC is located at the following address:

TX_UDC_Base + CH# × 16 (where CH# is the ATM channel number)

For receive channels in AAL2 UDC mode, the receiver copies the UDC header from the first cell received by the VC to the RX_UDC header table. The UDC headers of subsequent cells of that VC are discarded; UDC extended address mode (UEAD) is not affected.

The UDC header of a specific AAL2 receive VC is located at the following address:

RX_UDC_Base + CH# × 16 (where CH# is the ATM channel number)

The structure of a UDC header table (receive or transmit) is shown in Figure 42-23.

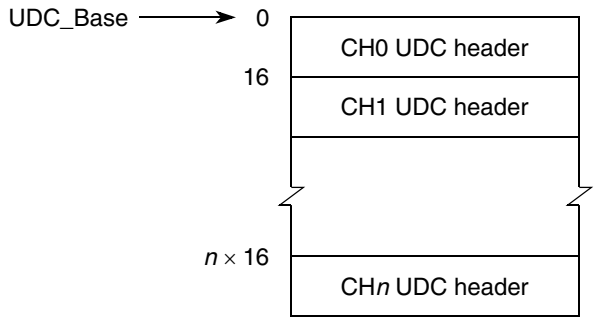


Figure 42-23. UDC Header Table

42.7 AAL2 Exceptions

For each VC, four circular interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ] for each VC, the user assigns an interrupt queue number.

When one of the CIDs generates an interrupt request, the CP writes a new entry to the interrupt queue containing the ATM channel number, the CID and a description of the exception. Because CID = 0 is a unique CID number, it is used to specify that the event is related to the VC rather than the CID. As with all ATM exceptions, the valid (V) bit is then set and INTQ_PTR is incremented. When INTQ_PTR reaches a location with the W bit set, it wraps to the first entry in the queue. More details can be found in Section 41.11, “ATM Exceptions.”

An interrupt entry for a CID is shown in Figure 42-24.

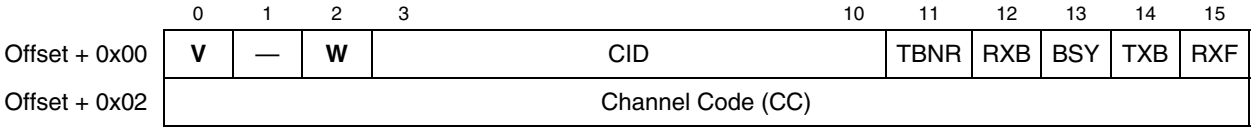


Figure 42-24. AAL2 Interrupt Queue Entry CID ≠ 0

Table 42-14 describes the interrupt queue entry fields for a CID.

Table 42-14. AAL2 Interrupt Queue Entry CID ≠ 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. The exception occurred for this CID.
	11	TBNR	Tx buffer not ready interrupt. This interrupt is issued when the CP tries to open a TxBD, which is not ready (R = 0). This interrupt is sent only if TxQD[BNM] = 1. The interrupt has an associated channel code and CID. **Note:** The CID number that is placed in the interrupt queue is the one currently located in the last BD. Because the CID is not updated when the BD is not ready, the CID value is the one extracted from this BD when it was last processed and transmitted. If the BD is never processed and the BD was cleared, the CID value could be zero.
	12	RXB[1]	Rx buffer interrupt. This interrupt is issued when the I bit is set for an RxBD and the RxQD[RBM] bit is set. This interrupt has an associated channel code and CID.
	13	BSY	Busy condition. The RxBD table associated with this channel's CID is busy. Packets were discarded due to this condition.
	14	TXB	Transmit buffer interrupt. This interrupt is issued when the TxBD[I] bit is set. This interrupt is sent only if TxQD[TBM] is set. This interrupt has an associated channel code and CID.
	15	RXF[1]	Receive SSSAR SDU (frame). An SSSAR frame belonging to this channel's CID has been received. This interrupt is sent only if RxQD[RFM] = 1.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

[1] These interrupt queue fields are defined differently for other AAL types. Refer toTable 41-41 for more information.

An interrupt entry for the VC is shown in Figure 42-25.

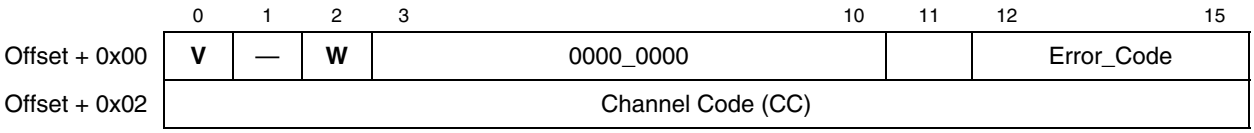


Figure 42-25. AAL2 Interrupt Queue Entry CID = 0

Table 42-15 describes the interrupt queue entry fields for the VC. All the receive error events are enabled by setting RCT[EM].

Table 42-15. AAL2 Interrupt Queue Entry CID = 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. Equals zero. This exception applies to the whole cell.
	11	—	Reserved
	12–15	Error_Code	A receive error was detected. 0000 Parity error of the OSF. 0001 The STF sequence number is incorrect. 0010 The number of octets expected to overlap into this cell does not match the OSF. 0011 OSF is greater than 47. 0100 A packet HEC error was detected. 0101 The length of the CPS packet exceeds the Max_SDU_Length. 0111 A packet HEC error was detected in a split header packet.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

Chapter 43
Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the MPC8555E to exchange data between other MPC8555E chips, the MPC860, the MC68360, the MC68302, the M68HC11 and M68HC05 microcontroller families, and peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock and slave select). The SPI block consists of transmitter and receiver sections, an independent baud-rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is sent and received simultaneously.

Because the SPI receiver and transmitter are double-buffered, as shown in Figure 43-1, the effective FIFO size (latency) is two characters. The SPI's msb is shifted out first. When the SPI is disabled in the SPI mode register (SPMODE[EN] = 0), it consumes little power.

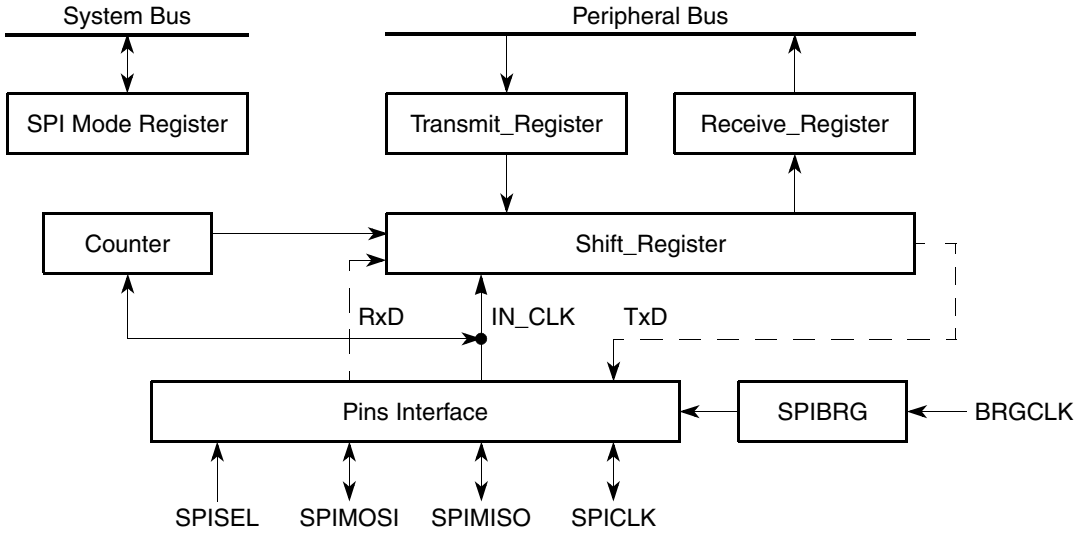


Figure 43-1. SPI Block Diagram

43.1 Features

The following is a list of the SPI's main features:

- Four-signal interface (SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$) multiplexed with port D signals
- Full-duplex operation
- Works with data characters from 4 to 16 bits long

- Supports back-to-back character transmission and reception
- Master or slave SPI modes supported
- Multiple-master environment support
- Continuous transfer mode for automatic scanning of a peripheral
- Supports maximum clock rates of 25 MHz in master mode and 50 MHz in slave mode, assuming a 100-MHz system clock
- Independent programmable baud rate generator
- Programmable clock phase and polarity
- Open-drain outputs support multiple-master configuration
- Local loopback capability for testing

43.2 SPI Clocking and Signal Functions

The SPI can be configured as a slave or as a master in single- or multiple-master environments. The master SPI generates the transfer clock SPICLK using the SPI baud rate generator (BRG). The SPI BRG takes its input from BRGCLK, which is generated in the MPC8555E clock synthesizer.

SPICLK is a gated clock, active only during data transfers. Four combinations of SPICLK phase and polarity can be configured with SPMODE[CI, CP]. SPI signals can also be configured as open-drain to support a multiple-master configuration in which a shared SPI signal is driven by the MPC8555E or an external SPI device.

The SPI master-in slave-out SPIMISO signal acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in SPIMOSI signal is an output for master devices and an input for slave devices. The dual functionality of these signals allows the SPIs in a multiple-master environment to communicate with one another using a common hardware configuration.

- When the SPI is a master, SPICLK is the clock output signal that shifts received data in from SPIMISO and transmitted data out to SPIMOSI. SPI masters must output a slave select signal to enable SPI slave devices by using a separate general-purpose I/O signal. Assertion of an SPI's $\overline{\text{SPISEL}}$ while it is master causes an error.
- When the SPI is a slave, SPICLK is the clock input that shifts received data in from SPIMOSI and transmitted data out through SPIMISO. $\overline{\text{SPISEL}}$ is the enable input to the SPI slave. In a multiple-master environment, $\overline{\text{SPISEL}}$ (always an input) is used to detect an error when more than one master is operating.

As described in Chapter 45, "Parallel I/O Ports," SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$ are multiplexed with port D[16:19] signals, respectively. They are configured as SPI signals through the port D signal assignment register (PDPAR) and the port D data direction register (PDDIR), specifically by setting PDPAR[DD*n*] and PDDIR[DR*n*].

43.3 Configuring the SPI Controller

The SPI can be programmed to work in a single- or multiple-master environment. This section describes SPI master and slave operation in a single-master configuration and then discusses the multi-master environment.

43.3.1 SPI as a Master Device

In master mode, the SPI sends a message to the slave peripheral, which sends back a simultaneous reply. A single master MPC8555E with multiple slaves can use general-purpose parallel I/O signals to selectively enable slaves, as shown in Figure 43-2. To eliminate the multiple-master error in a single-master environment, the master's $\overline{\text{SPISEL}}$ input can be forced inactive by selecting port D[19] for general-purpose I/O (PDPAR[DD19] = 0).

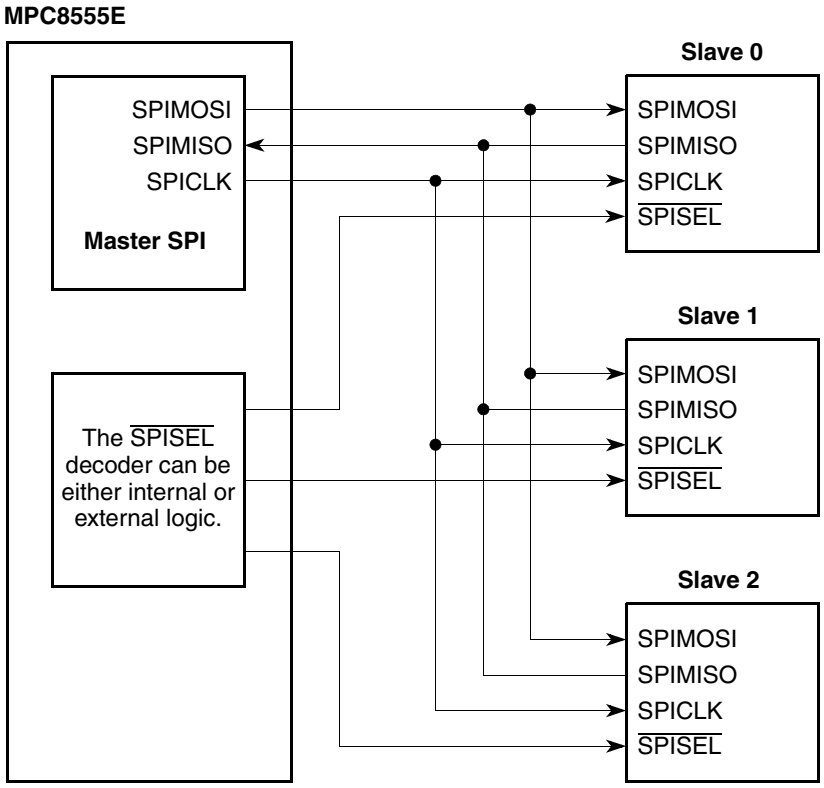


Figure 43-2. Single-Master/Multi-Slave Configuration

To start exchanging data, the core writes the data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core then sets SPCOM[STR] in the SPI command register to start sending data, which starts once the SDMA channel loads the Tx FIFO with data.

The SPI then generates programmable clock pulses on SPICLK for each character and simultaneously shifts Tx data out on SPIMOSI and Rx data in on SPIMISO. Received data is written into a Rx buffer using the next available RxBD. The SPI keeps sending and receiving characters until the whole buffer is sent or an error occurs. The CP then clears TxBD[R] and RxBD[E] and issues a maskable interrupt to the interrupt controller in the SIU.

When multiple TxBDs are ready, TxBD[L] determines whether the SPI keeps transmitting without SPCOM[STR] being set again. If the current TxBD[L] is cleared, the next TxBD is processed after data from the current buffer is sent. Typically there is no delay on SPIMOSI between buffers. If the current TxBD[L] is set, sending stops after the current buffer is sent. In addition, the RxBD is closed after transmission stops, even if the Rx buffer is not full; therefore, Rx buffers need not be the same length as Tx buffers.

43.3.2 SPI as a Slave Device

In slave mode, the SPI receives messages from an SPI master and sends a simultaneous reply. The slave's $\overline{\text{SPISEL}}$ must be asserted before Rx clocks are recognized; once $\overline{\text{SPISEL}}$ is asserted, SPICLK becomes an input from the master to the slave. SPICLK can be any frequency from DC to BRGCLK/2 (12.5 MHz for a 25-MHz system).

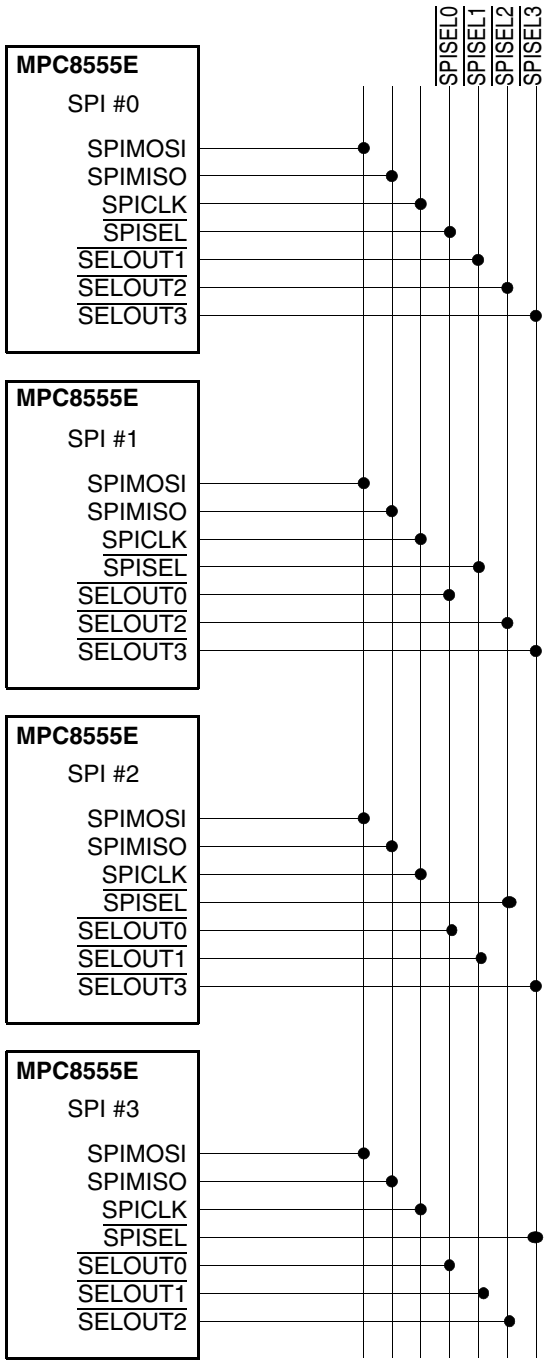
To prepare for data transfers, the slave's core writes data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core then sets SPCOM[STR] to activate the SPI. Once $\overline{\text{SPISEL}}$ is asserted, the slave shifts data out from SPIMISO and in through SPIMOSI. A maskable interrupt is issued when a full buffer finishes receiving and sending or after an error. The SPI uses successive RxBDs in the table to continue reception until it runs out of Rx buffers or $\overline{\text{SPISEL}}$ is negated.

Transmission continues until no more data is available or $\overline{\text{SPISEL}}$ is negated. If it is negated before all data is sent, it stops but the TxBD stays open. Transmission continues once $\overline{\text{SPISEL}}$ is reasserted and SPICLK begins toggling. After the characters in the buffer are sent, the SPI sends ones as long as $\overline{\text{SPISEL}}$ remains asserted.

43.3.3 SPI in Multiple-Master Operation

The SPI can operate in a multiple-master environment in which SPI devices are connected to the same bus. In this configuration, the SPIMOSI, SPIMISO, and SPICLK signals of all SPIs are shared; the $\overline{\text{SPISEL}}$ inputs are connected separately, as shown in Figure 43-3. Only one SPI device can act as master at a time—all others must be slaves. When an SPI is configured as a master and its $\overline{\text{SPISEL}}$ input is asserted, a multiple-master error occurs because more than one SPI device is a bus master. The SPI sets SPIE[MME] in the SPI event register and a maskable interrupt is issued to the core. It also disables SPI operation and the output drivers of SPI signals. The core must clear SPMODE[EN] before the SPI is used again. After correcting the problems, clear SPIE[MME] and re-enable the SPI.

The maximum sustained data rate that the SPI supports is SYSTEMCLK/50. However, the SPI can transfer a single character at much higher rates—SYSTEMCLK/4 in master mode and SYSTEMCLK/2 in slave mode. Gaps should be inserted between multiple characters to keep from exceeding the maximum sustained data rate.



Notes:
- All signals are open-drain.
- For a system with more than two masters, $\overline{\text{SPISEL}}$ and SPIE[MME] do not detect all possible conflicts
- It is the responsibility of software to arbitrate for the SPI bus (with token passing, for example).
- $\overline{\text{SELOUT}x}$ signals are implemented in software with general-purpose I/O signals.

Figure 43-3. Multiple-Master Configuration

43.4 Programming the SPI Registers

The following sections describe the registers used in configuring and operating the SPI.

43.4.1 SPI Mode Register (SPMODE)

The SPI mode register (SPMODE), shown in Figure 43-4, controls both the SPI operation mode and clock source.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—	LOOP	CI	CP	DIV16	REV	M/S	EN	LEN				PM			
Reset	0000_00						—	0_0000_0000								
R/W	R/W															
Offset	0x0x9_1AA0															

Figure 43-4. SPMODE—SPI Mode Register

Table 43-1 describes SPMODE fields.

Table 43-1. SPMODE Field Descriptions

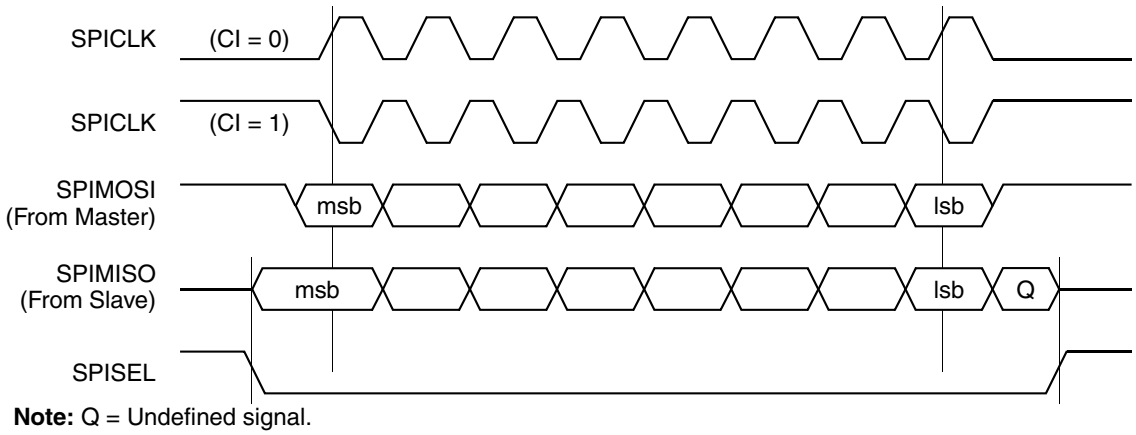
Bits	Name[1]	Description
0	—	Reserved, should be cleared.
1	**LOOP**	Loop mode. Enables local loopback operation. 0 Normal operation 1 Loopback mode. The transmitter output is internally connected to the receiver input. The receiver and transmitter operate normally, except that received data is ignored.
2	**CI**	Clock invert. Inverts SPI clock polarity. See Figure 43-5 and Figure 43-6. 0 The inactive state of SPICLK is low. 1 The inactive state of SPICLK is high.
3	**CP**	Clock phase. Selects the transfer format. See Figure 43-5 and Figure 43-6. 0 SPICLK starts toggling in the middle of the data transfer. 1 SPICLK starts toggling at the beginning of the data transfer.
4	**DIV16**	Divide by 16. Selects the clock source for the SPI baud rate generator when configured as an SPI master. In slave mode, SPICLK is the clock source. 0 BRGCLK is the input to the SPI BRG. 1 BRGCLK/16 is the input to the SPI BRG.
5	**REV**	Reverse data. Determines the receive and transmit character bit order. 0 Reverse data—lsb of the character sent and received first. 1 Normal operation—msb of the character sent and received first.
6	**M/S**	Master/slave. Selects master or slave mode. 0 The SPI is a slave. 1 The SPI is a master.
7	**EN**	Enable SPI. Do not change other SPMODE bits when EN is set. 0 The SPI is disabled. The SPI is in a reset state and consumes minimal power. The SPI BRG is not functioning and the input clock is disabled. 1 The SPI is enabled. Configure SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$ to connect to the SPI as described in Section 45.2, “Port Registers.”

Table 43-1. SPMODE Field Descriptions (continued)

Bits	Name[1]	Description
8–11	**LEN**	Character length in bits per character. If the character length is not greater than a byte, every byte in memory holds (LEN+1) valid bits. If the character length is greater than a byte, every half-word holds (LEN+1) valid bits. See Section 43.4.1.1, "SPI Examples with Different SPMODE[LEN] Values." 0000–0010Reserved, causes erratic behavior. 0011 4-bit characters … 1111 16-bit characters
12–15	**PM**	Prescale modulus select. Specifies the divide ratio of the prescale divider in the SPI clock generator. BRGCLK is divided by 4 × ([PM0–PM3] + 1), a range from 4 to 64. The clock has a 50% duty cycle.

[1] **Boldfaced** entries must be initialized by the user.

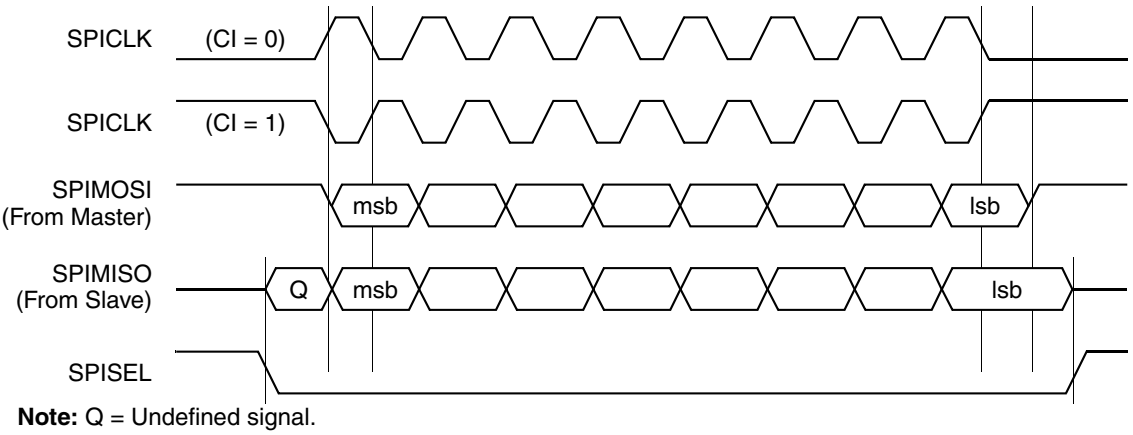
Figure 43-5 shows the SPI transfer format in which SPICLK starts toggling in the middle of the transfer (SPMODE[CP] = 0).



Note: Q = Undefined signal.

Figure 43-5. SPI Transfer Format with SPMODE[CP] = 0

Figure 43-6 shows the SPI transfer format in which SPICLK starts toggling at the beginning of the transfer (SPMODE[CP] = 1).



Note: Q = Undefined signal.

Figure 43-6. SPI Transfer Format with SPMODE[CP] = 1

43.4.1.1 SPI Examples with Different SPMODE[LEN] Values

The examples below show how SPMODE[LEN] is used to determine character length. To help map the process, the conventions shown in Table 43-2 are used in the examples.

Table 43-2. Example Conventions

Convention	Description
g–v	Binary symbols
x	Deleted bit
__ [1]	Original byte boundary
_ [1]	Original 4-bit boundary

[1] Both __ and _ are used to aid readability.

Once the data string image is determined, it is always transmitted byte by byte with the lsb of the most-significant byte sent first. For all examples below, assume the memory contains the following binary image:

```
        msb     ghij_klmn__opqr_stuv       lsb
```

Example 1

```
with LEN=4 (data size=5), the following data is selected:
        msb     xxxj_klmn__xxxr_stuv       lsb
with REV=0, the data string image is:
        msb     j_klmn__r_stuv             lsb
the order of the string appearing on the line, a byte at a time is:
        first   nmlk_j__vuts_r             last
with REV=1,the string has each byte reversed, and the data string image is:
        msb     nmlk_j__vuts_r             lsb
the order of the string appearing on the line, one byte at a time is:
        first   j_klmn__r_stuv             last
```

Example 2

```
with LEN=7 (data size=8), the following data is selected:
        msb     ghij_klmn__opqr_stuv       lsb
the data string is selected:
        msb     ghij_klmn__opqr_stuv       lsb
with REV=0, the string transmitted, a byte at a time with lsb first is:
        first   nmlk_jihg__vuts_rqpo       last
w/ REV=1, the string is byte reversed & transmitted, a byte at a time, lsb first:
        first   ghij_klmn__opqr_stuv       last
```

Example 3

```
with LEN=0xC (data size=13), the following data is selected:
        msb     ghij_klmn__xxxr_stuv       lsb
the data string selected is:
        msb     r_stuv__ghij_klmn          lsb
with REV=0, the string transmitted, a byte at a time with lsb first is:
        first   vuts_r__nmlk_jihg          last
with REV=1, the string is half-word reversed:
        msb     nmlk_jihg__vuts_r          lsb
and transmitted a byte at a time with lsb first:
        first   ghij_klmn__r_stuv          last
```

43.4.2 SPI Event/Mask Registers (SPIE/SPIM)

The SPI event register (SPIE) generates interrupts and reports events recognized by the SPI. When an event is recognized, the SPI sets the corresponding SPIE bit. Clear SPIE bits by writing a 1—writing 0 has no effect. Setting a bit in the SPI mask register (SPIM) enables and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the CP clears internal interrupt requests. Figure 43-7 shows both registers.

	0	1	2	3	4	5	6	7
Field	—		MME	TXE	—	BSY	TXB	RXB
Reset	0000_0000							
R/W	R/W							
Offset	0x0x9_1AA6 (SPIE); 0x0x9_1AAA (SPIM)							

Figure 43-7. SPIE/SPIM—SPI Event/Mask Registers

Table 43-3 describes the SPIE/SPIM fields.

Table 43-3. SPIE/SPIM Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	MME	Multiple-master error. Set when $\overline{\text{SPISEL}}$ is asserted externally while the SPI is in master mode.
3	TXE	Tx error. Set when an error occurs during transmission.
4	—	Reserved, should be cleared.
5	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.
6	TXB	Tx buffer. Set when the Tx data of the last character in the buffer is written to the Tx FIFO. Wait two character times to be sure data is completely sent over the transmit signal.
7	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the BD is closed.

43.4.3 SPI Command Register (SPCOM)

The SPI command register (SPCOM), shown in Figure 43-8, is used to start SPI operation.

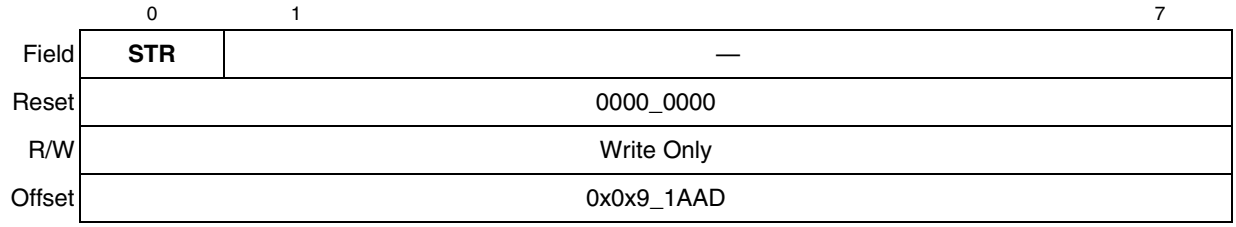


Figure 43-8. SPCOM—SPI Command Register

Table 43-4 describes the SPCOM fields.

Table 43-4. SPCOM Field Descriptions

Bits	Name[1]	Description
0	**STR**	Start transmit. For an SPI master, setting STR causes the SPI to start transferring data to and from the Tx/Rx buffers if they are prepared. For a slave, setting STR when the SPI is idle causes it to load the Tx data register from the SPI Tx buffer and start sending with the next SPICLK after $\overline{\text{SPISEL}}$ is asserted. STR is cleared automatically after one system clock cycle.
1–7	—	Reserved and should be cleared.

[1] **Boldfaced** entries must be initialized by the user.

43.5 SPI Parameter RAM

The SPI parameter RAM area is similar to the SCC general-purpose parameter RAM. The CP accesses the SPI parameter table using a user-programmed pointer (SPI_BASE) located in the parameter RAM; see Section 21.4.2, "Parameter RAM." The SPI parameter table can be placed at any 64-byte aligned address in the dual-port RAM's general-purpose area (banks 1–8, 11 and 12). Some parameter values must be user-initialized before the SPI is enabled; the CP initializes the others. Once initialized, parameter RAM values do not usually need to be accessed. They should be changed only when the SPI is inactive. Table 43-5 shows the memory map of the SPI parameter RAM.

Table 43-5. SPI Parameter RAM Memory Map

Offset [1]	Name[2]	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address. Indicate where the BD tables begin in the dual-port RAM. Setting Rx/TxBD[W] in the last BD in each BD table determines how many BDs are allocated for the Tx and Rx sections of the SPI. Initialize RBASE/TBASE before enabling the SPI. Furthermore, do not configure BD tables of the SPI to overlap any other active controller's parameter RAM. RBASE and TBASE should be divisible by eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code registers. The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. See Section 43.5.1, "Receive/Transmit Function Code Registers (RFCR/TFCR)."
0x05	**TFCR**	Byte	

Table 43-5. SPI Parameter RAM Memory Map (continued)

Offset [1]	Name[2]	Width	Description
0x06	**MRBLR**	Hword	Maximum receive buffer length. The SPI has one MRBLR entry to define the maximum number of bytes the MPC8555E writes to a Rx buffer before moving to the next buffer. The MPC8555E can write fewer bytes than MRBLR if an error or end-of-frame occurs, but never exceeds the MRBLR value. User-supplied buffers should be no smaller than MRBLR. Tx buffers are unaffected by MRBLR and can have varying lengths; the number of bytes to be sent is programmed in TxBD[Data Length]. MRBLR is not intended to be changed while the SPI is operating. However it can be changed in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). The change takes effect when the CP moves control to the next RxBD. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the SPI receiver is disabled. MRBLR should be greater than zero; it should be an even number if the character length of the data exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state.[3] Reserved for CP use.
0x0C	—	Word	The Rx internal data pointer [3] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the current RxBD being processed or to the next BD to be serviced when idle. After a reset or when the end of the BD table is reached, the CP initializes RBPTR to the RBASE value. Most applications should not modify RBPTR, but it can be updated when the receiver is disabled or when no Rx buffer is in use.
0x12	—	Hword	The Rx internal byte count [3] is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	—	Word	Rx temp.[3] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state.[3] Reserved for CP use.
0x1C	—	Word	The Tx internal data pointer[3] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the current TxBD during frame transmission or the next BD to be processed when idle. After reset or when the end of the TxBD table is reached, the CP initializes TBPTR to the TBASE value. Most applications do not need to modify TBPTR, but it can be updated when the transmitter is disabled or when no Tx buffer is in use.
0x22	—	Hword	The Tx internal byte count[3] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.
0x24	—	Word	Tx temp.[3] Reserved for CP use.
0x34	—	Word	SDMA temp.

1 From the pointer value programmed in SPI_BASE at CCSRBAR + 0x89FC.

2 **Boldfaced** entries must be initialized by the user.

3 Normally, these parameters need not be accessed. They are listed to help experienced users in debugging.

43.5.1 Receive/Transmit Function Code Registers (RFCR/TFCR)

Figure 43-9 shows the fields in the receive/transmit function code registers (RFCR/TFCR).

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**TC2**	**DTB**	—
Reset	0000_0000							
R/W	R/W							
Offset	SPI Base + 04 (RFCR)/SPI Base + 05 (TFCR)							

Figure 43-9. RFCR/TFCR—Function Code Registers

Table 43-6 describes the RFCR/TFCR fields.

Table 43-6. RFCR/TFCR Field Descriptions

Bits	Name[1]	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global access bit 0 Disable memory snooping 1 Enable memory snooping
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame or BD. 00 True little endian. Note this mode can only be used with 32-bit port size memory. 01 Big endian 1x Munged little endian
5	**TC2**	Transfer code 2. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Data bus indicator. 0 Use system bus for SDMA operation. 1 Reserved.
7	—	Reserved, should be cleared.

[1] **Boldfaced** entries must be initialized by the user.

43.6 SPI Commands

Table 43-7 lists transmit/receive commands sent to the CP command register (CPCR).

Table 43-7. SPI Commands

Command	Description
INIT TX PARAMETERS	Initializes all transmit parameters in the parameter RAM to their reset state and should be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

Table 43-7. SPI Commands (continued)

Command	Description
CLOSE RXBD	Forces the SPI controller to close the current RxBD and use the next BD for subsequently received data. If the controller is not receiving data, no action is taken. Use this command to extract data from a partially full buffer.
INIT RX PARAMETERS	Initializes all receive parameters in the parameter RAM to their reset state. Should be issued only when the receiver is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

43.7 SPI Buffer Descriptor (BD) Table

As shown in Figure 43-10, BDs are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as for the SCCs and SMCs and form circular queues that determine the order buffers are transferred. The CP uses BDs to confirm reception and transmission or to indicate error conditions so that the core knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the dual-port RAM.

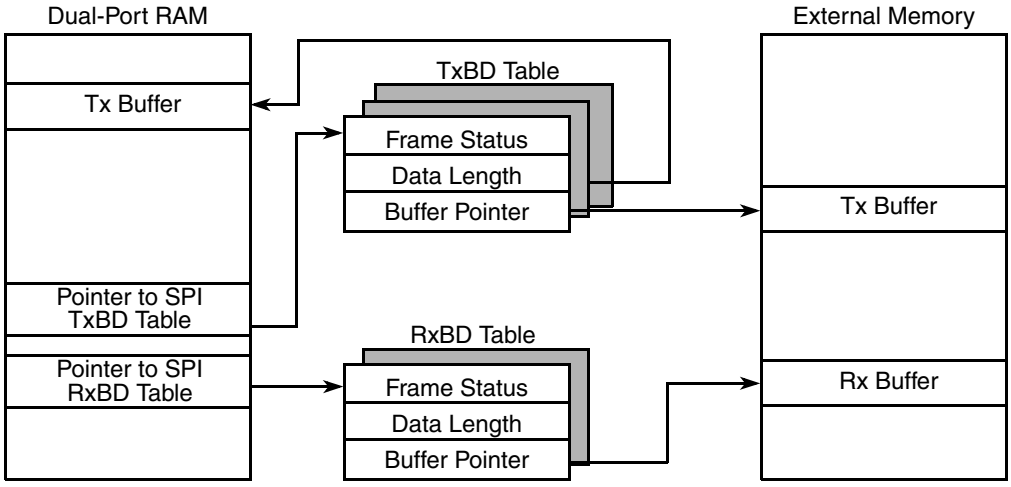


Figure 43-10. SPI Memory Structure

43.7.1 SPI Buffer Descriptors (BDs)

Receive and transmit BDs report information about each buffer transferred and whether a maskable interrupt should be generated. Each 64-bit BD, shown in Figure 43-11 and Figure 43-12, has the following structure:

- The half word at offset + 0 contains status and control bits. The CP updates the status bits after the buffer is sent or received.
- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
  - For an RxBD, this is the number of octets the CP writes into this RxBD's buffer once the BD closes. The CP updates this field after the received data is placed into the buffer. Memory allocated for this buffer should be no smaller than MRBLR.
  - For a TxBD, this is the number of octets the CP should transmit from its buffer. Normally, this value should be greater than zero. If the character length is more than 8 bits, the data length

should be even. For example, to send three characters of 8-bit data, 1 start, and 1 stop, the data length field should be initialized to 3. However, to send three characters of 9-bit data, the data length field should be initialized to 6 since the three 9-bit data fields occupy three half-words in memory. The CP never modifies this field.

- The word at offset + 4 points to the beginning of the buffer.
  - For an RxBD, the pointer must be even and can point to internal or external memory.
  - For a TxBD, the pointer can be even or odd, unless the character exceeds 8 bits, for which it must be even. The buffer can be in internal or external memory.

43.7.1.1 SPI Receive BD (RxBD)

The CP uses RxBDs to report on each received buffer. It closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer once the current buffer is full. The CP also closes the buffer when the SPI is configured as a slave and $\overline{\text{SPISEL}}$ is negated, indicating that reception stopped. The core should write RxBD bits before the SPI is enabled. The format of an RxBD is shown in Figure 43-11.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	—	**CM**	—							OV	ME
Offset + 2	Data Length															
Offset + 4	**Rx Buffer Pointer**															
Offset + 6																

Figure 43-11. SPI RxBD

Table 43-8 describes the RxBD status and control fields.

Table 43-8. SPI RxBD Status and Control Field Descriptions

Bits	Name[1]	Description
0	E	Empty. 0 The buffer is full or stopped receiving because of an error. The core can examine or write to any fields of this RxBD, but the CP does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	W	Wrap (last BD in table). 0 Not the last BD in the RxBD table. 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 SPIE[RXB] is set when this buffer is full, indicating the need for the core to process the buffer. SPIE[RXB] causes an interrupt if not masked.
4	L	Last. Updated by the SPI when the buffer is closed because $\overline{\text{SPISEL}}$ was negated (slave mode only). Otherwise, RxBD[ME] is set. The SPI updates L after received data is placed in the buffer. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message.

Table 43-8. SPI RxBD Status and Control Field Descriptions (continued)

Bits	Name[1]	Description
5	—	Reserved, should be cleared.
6	CM	Continuous mode. Master mode only; in slave mode, CM should be cleared. 0 Normal operation. 1 The CP does not clear RxBD[E] after this BD is closed; the buffer is overwritten when the CP next accesses this BD. This allows continuous reception from an SPI slave into one buffer for autoscanning of a serial A/D peripheral with no core overhead.
7–13	—	Reserved, should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception (slave mode only). The SPI updates OV after the received data is placed in the buffer.
15	ME	Multiple-master error. Set when this buffer is closed because $\overline{\text{SPISEL}}$ was asserted when the SPI was in master mode. Indicates a synchronization problem between multiple masters on the SPI bus. The SPI updates ME after the received data is placed in the buffer.

[1] **Boldfaced** entries must be initialized by the user.

43.7.1.2 SPI Transmit BD (TxBD)

Data to be sent with the SPI is sent to the CP by arranging it in buffers referenced by TxBDs in the TxBD table. TxBD fields should be prepared before data is sent. The format of an TxBD is shown in Figure 43-12.

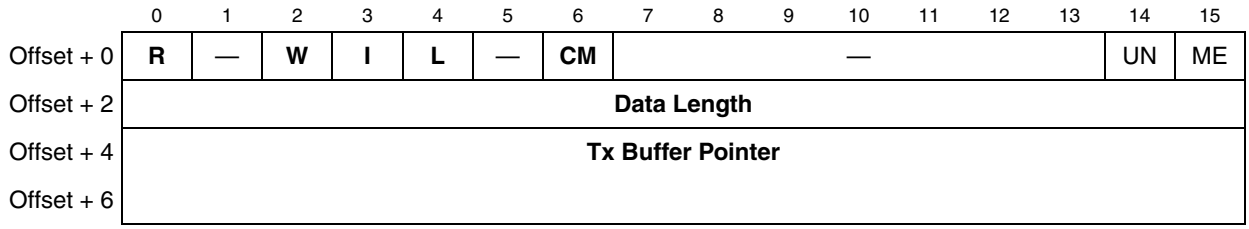


Figure 43-12. SPI TxBD

Table 43-9 describes the TxBD status and control fields.

Table 43-9. SPI TxBD Status and Control Field Descriptions

Bits	Name[1]	Description
0	**R**	Ready 0 The buffer is not ready to be sent. This BD or its buffer can be modified. The CP clears R (unless RxBD[CM] is set) after the buffer is sent (unless RxBD[CM] is set) or an error occurs. 1 The buffer is ready for transmission or is being sent. The BD cannot be modified once R is set.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in TxBD table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data using the BD pointed to by TBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.

Table 43-9. SPI TxBD Status and Control Field Descriptions (continued)

Bits	Name[1]	Description
3	**I**	Interrupt 0 No interrupt is generated after this buffer is processed. 1 SPIE[TXB] or SPIE[TXE] are set when this buffer is processed and causes interrupts if not masked.
4	**L**	Last 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message.
5	—	Reserved, should be cleared.
6	**CM**	Continuous mode. Valid only when the SPI is in master mode. In slave mode, it should be cleared. 0 Normal operation 1 The CP does not clear TxBD[R] after this BD is closed, allowing the buffer to be resent automatically when the CP next accesses this BD.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. Indicates that the SPI encountered a transmitter underrun condition while sending the buffer. This error occurs only when the SPI is in slave mode. The SPI updates UN after it sends the buffer.
15	ME	Multiple-master error. Indicates that this buffer is closed because $\overline{\text{SPISEL}}$ was asserted when the SPI was in master mode. A synchronization problem occurred between devices on the SPI bus. The SPI updates ME after sending the buffer.

[1] **Boldfaced** entries must be initialized by the user.

43.8 SPI Master Programming Example

The following sequence initializes the SPI to run at a high speed in master mode:

1. Configure port D to enable SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$.
2. Configure a parallel I/O signal to operate as the SPI select output signal if needed.
3. In address 0x89FC, assign a pointer to the SPI parameter RAM.
4. Write RBASE and TBASE in the SPI parameter RAM to point to the RxBD and TxBD tables in the dual-port RAM. Assuming one RxBD followed by one TxBD at the beginning of the dual-port RAM, write RBASE with 0x0000 and TBASE with 0x0008.
5. Write RFCR and TFCR with 0x10 for normal operation.
6. Write MRBLR with the maximum number of bytes per Rx buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
7. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
8. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
9. Execute the INIT RX AND TX PARAMETERS command by writing 0x2541_0000 to CPCR.
10. Write 0xFF to SPIE to clear any previous events.
11. Write 0x37 to SPIM to enable all possible SPI interrupts.

12. Write 0x0370 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, 8-bit characters, and the fastest speed possible.
13. Set SPCOM[STR] to start the transfer.

After 5 bytes are sent, the TxBD is closed. Additionally, the Rx buffer is closed after 5 bytes are received because TxBD[L] is set.

43.9 SPI Slave Programming Example

The following is an example initialization sequence to follow when the SPI is in slave mode. It is very similar to the SPI master example, except that $\overline{\text{SPISEL}}$ is used instead of a general-purpose I/O signal (as shown in Figure 43-2).

1. Enable SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$.
2. In address 0x89FC, assign a pointer to the SPI parameter RAM.
3. Assuming one RxBD at the beginning of the dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008 in the SPI parameter RAM.
4. Write RFCR and TFCR with 0x10 for normal operation.
5. Program MRBLR = 0x0010 for 16 bytes, the maximum number of bytes per buffer.
6. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
7. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
8. Execute the INIT RX AND TX PARAMETERS command by writing 0x2541_0000 to CPCR.
9. Write 0xFF to SPIE to clear any previous events.
10. Write 0x37 to SPIM to enable all SPI interrupts.
11. Set SPMODE to 0x0170 to enable normal operation (not loopback), slave mode, SPI enabled, and 8-bit characters. BRG speed is ignored in slave mode.
12. Set SPCOM[STR] to enable the SPI to be ready once the master begins the transfer.

NOTE

If the master sends 3 bytes and negates $\overline{\text{SPISEL}}$, the RxBD is closed but the TxBD remains open. If the master sends 5 or more bytes, the TxBD is closed after the fifth byte. If the master sends 16 bytes and negates $\overline{\text{SPISEL}}$, the RxBD is closed without triggering an out-of-buffers error. If the master sends more than 16 bytes, the RxBD is closed (full) and an out-of-buffers error occurs after the 17th byte is received.

43.10 Handling Interrupts in the SPI

The following sequence should be followed to handle interrupts in the SPI:

1. Once an interrupt occurs, read SPIE to determine the interrupt source. Normally, SPIE bits should be cleared at this time.
2. Process the TxBD to reuse it and the RxBD to extract the data from it. To transmit another buffer, simply set TxBD[R], RxBD[E], and SPCOM[STR].
3. Execute an **rfi** instruction.

Chapter 44
I²C Controller

The inter-integrated circuit (I²C) controller lets the MPC8555E exchange data with other I²C devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCD displays. The I²C controller uses a synchronous, multiple-master bus that can connect several integrated circuits on a board. It uses two signals—serial data (SDA) and serial clock (SCL)—to carry information between the integrated circuits connected to it.

As shown in Figure 44-1, the I²C controller consists of transmit and receive sections, an independent baud-rate generator (BRG), and a control unit. The transmit and receive sections use the same clock, which is derived from the I²C BRG when in master mode and generated externally when in slave mode. Wait states are inserted during a data transfer if SCL is held low by a slave device. In the middle of a data transfer, the master I²C controller recognizes the need for wait states by monitoring SCL. However, the I²C controller has no automatic time-out mechanism if the slave device does not release SCL; therefore, software should monitor how long SCL stays low to generate bus timeouts.

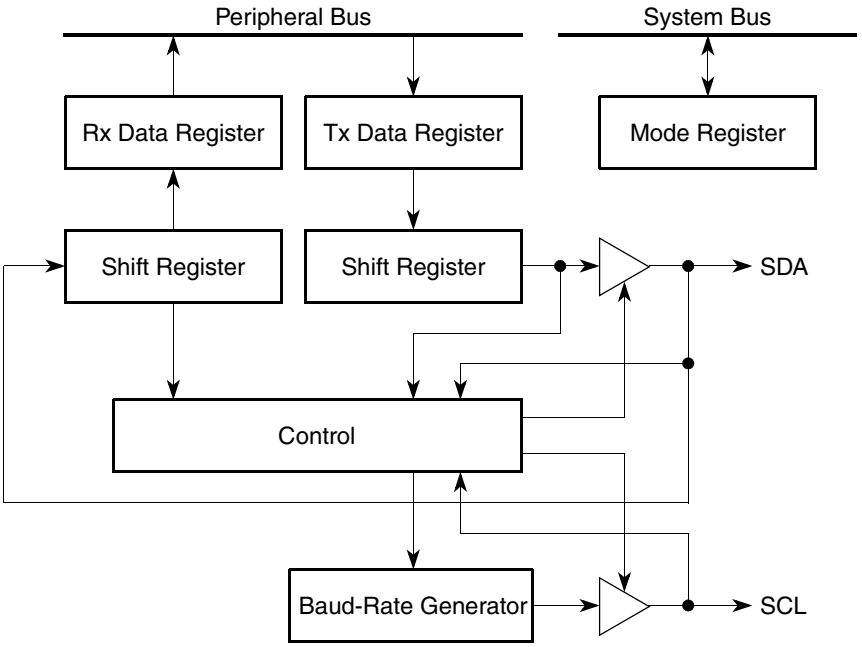


Figure 44-1. I²C Controller Block Diagram

The I²C receiver and transmitter are double-buffered, which corresponds to an effective two-character FIFO latency. In normal operation, the transmitter shifts the msb (bit 0) out first. When the I²C is not enabled in the I²C mode register (I2MOD[EN] = 0), it consumes little power.

44.1 Features

The following is a list of the I²C controller's main features:

- Two-signal interface (SDA and SCL)
- Support for master and slave I²C operation
- Multiple-master environment support
- Continuous transfer mode for automatic scanning of a peripheral
- Supports a maximum clock rate of 2080 kHz (with a CPM utilization of 25%), assuming a 100-MHz system clock.
- Independent, programmable baud-rate generator
- Supports 7-bit I²C addressing
- Open-drain output signals allow multiple master configuration
- Local loopback capability for testing

44.2 I²C Controller Clocking and Signal Functions

The I²C controller can be configured as a master or slave for the serial channel. As a master, the controller's BRG provides the transfer clock. The I²C BRG takes its input from the BRG clock (BRGCLK), which is generated from the CPM clock.

SDA and SCL are bidirectional signals connected to a positive supply voltage through an external pull-up resistor. When the bus is free, both signals are pulled high. The general I²C master/slave configuration is shown in Figure 44-2.

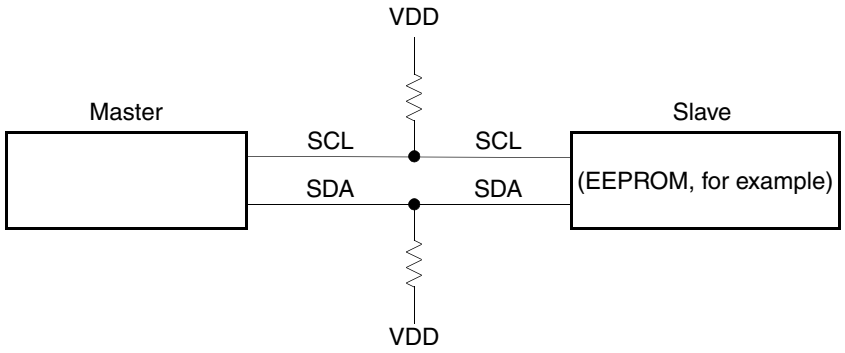


Figure 44-2. I²C Master/Slave General Configuration

When the I²C controller is master, the SCL clock output, taken directly from the I²C BRG, shifts receive data in and transmit data out through SDA. The transmitter arbitrates for the bus during transmission and aborts if it loses arbitration. When the I²C controller is a slave, the SCL clock input shifts data in and out through SDA. The SCL frequency can range from DC to BRGCLK/48.

44.3 I²C Controller Transfers

To initiate a transfer, the master I²C controller sends a message specifying a read or write request to an I²C slave. The first byte of the message consists of a 7-bit slave port address and a R/W request bit. Note that

because the R/W request follows the slave port address in the I²C bus specification, the R/W request bit must be placed in the lsb (bit 7) unless operating in reverse data mode; see Section 44.4.1, "I²C Mode Register (I2MOD)."

To write to a slave, the master sends a write request (R/W = 0) along with either the target slave's address or a general call (broadcast) address of all zeros, followed by the data to be written. To read from a slave, the master sends a read request (R/W = 1) and the target slave's address. When the target slave acknowledges the read request, the transfer direction is reversed, and the master receives the slave's transmit buffer(s). If the receiver (master or slave) does not acknowledge each byte transfer in the ninth bit frame, the transmitter signals a transmission error event (I2ER[TXE]). An I²C transfer timing diagram is shown in Figure 44-3.

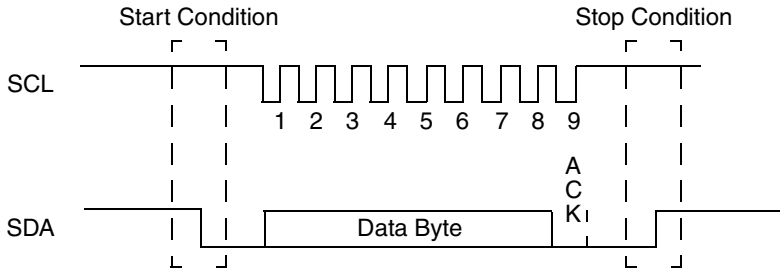


Figure 44-3. I²C Transfer Timing

Select master or slave mode for the controller using the I²C command register (I2COM[M/S]). Set the master's start bit, I2COM[STR], to begin a transfer; setting a slave's I2COM[STR] activates the slave to wait for a transfer request from a master.

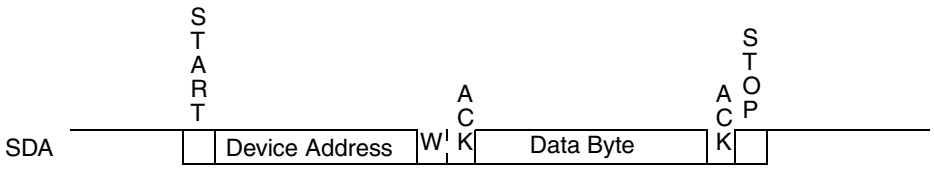
If a master or slave transmitter's current TxBD[L] is set, transmission stops once the buffer is sent; that is, I2COM[STR] must be set again to reactivate transfers. If TxBD[L] is zero, once the current buffer is sent, the controller begins processing the next TxBD without waiting for I2COM[STR] to be set again.

The following sections further detail the transfer process.

44.3.1 I²C Master Write (Slave Read)

If the MPC8555E is the master, prepare the transmit buffers and BDs before initiating a write. Initialize the first transmit data byte with the slave address and write request (R/W = 0).

If the MPC8555E is the slave target of the write, prepare receive buffers and BDs to await the master's request. Figure 44-4 shows the timing for a master write.



Note: Data and ACK are repeated n times.

Figure 44-4. I²C Master Write Timing

A master write occurs as follows:

1. The master core sets I2COM[STR]. The transfer starts when the SDMA channel loads the Tx FIFO with data and the I²C bus is not busy.
2. The I²C master generates a start condition—a high-to-low transition on SDA while SCL is high—and the transfer clock SCL pulses for each bit shifted out on SDA. If the master transmitter detects a multiple-master collision (by sensing a '0' on SDA while sending a '1'), transmission stops and the channel reverts to slave mode. A maskable interrupt is sent to the master's core so software can try to retransmit later.
3. The slave acknowledges each byte and writes to its current receive buffer until a new start or stop condition is detected.
4. After sending each byte, the master monitors the acknowledge indication. If the slave receiver fails to acknowledge a byte, transmission stops and the master generates a stop condition—a low-to-high transition on SDA while SCL is high.

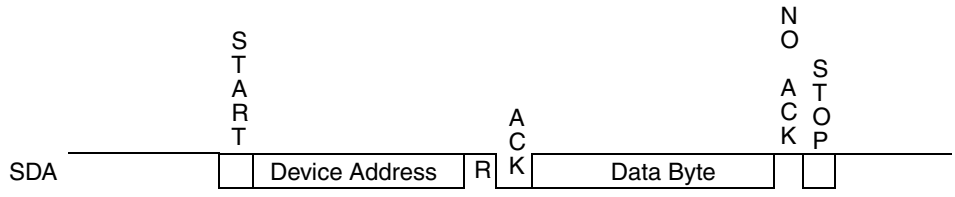
44.3.2 I²C Loopback Testing

When in master mode, an I²C controller supports loopback operation for master write requests. The master I²C controller simply issues a write request directed to its own address (programmed in I2ADD). The master's receiver monitors the transmission and reads the transmitted data into its receive buffer. Loopback operation requires no special register programming.

44.3.3 I²C Master Read (Slave Write)

Before initiating a master read with the MPC8555E, prepare a transmit buffer of size *n* + 1 bytes, where *n* is the number of bytes to be read from the slave. The first transmit byte should be initialized to the slave address with R/W = 1. The next *n* transmit bytes are used strictly for timing and can be left uninitialized. Configure suitable receive buffers and BDs to receive the slave's transmission.

If the MPC8555E is the slave target of the read, prepare the I²C transmit buffers and BDs and activate it by setting I2COM[STR]. Figure 44-5 shows the timing for a master read.



Note: After the nth data byte, the master does not acknowledge the slave.

Figure 44-5. I²C Master Read Timing

A master read occurs as follows:

1. Set the master's I2COM[STR] to initiate the read. The transfer starts when the SDMA channel loads the transmit FIFO with data and the I²C bus is not busy.
2. The slave detects a start condition on SDA and SCL.

3. After the first byte is shifted in, the slave compares the received data to its slave address. If the slave is an MPC8555E, the address is programmed in its I²C address register (I2ADD).
   — If a match is found, the slave acknowledges the received byte and begins transmitting on the clock pulse immediately following the acknowledge.
   — If a match is found but the slave is not ready, the read request is not acknowledged and the transaction is aborted. If the slave is an MPC8555E, a maskable transmission error interrupt is triggered to allow software to prepare data for transmission on the next try.
   — If a mismatch occurs, the slave ignores the message and searches for a new start condition.
4. The master acknowledges each byte sent as long as an overrun does not occur. If the master receiver fails to acknowledge a byte, the slave aborts transmission. For a slave MPC8555E, the abort generates a maskable interrupt. A maskable interrupt is also issued after a complete buffer is sent or after an error. If an underrun occurs, the MPC8555E slave sends ones until a stop condition is detected.

44.3.4 I²C Multi-Master Considerations

The I²C controller supports a multi-master configuration, in which the I²C controller must alternate between master and slave modes. The I²C controller supports this by implementing I²C master arbitration in hardware. However, due to the nature of the I²C bus and the implementation of the I²C controller, certain software considerations must be made.

A MPC8555E I²C controller attempting a master read request could simultaneously be targeted for an external master write (slave read). Both operations trigger the controller's I2CER[RXB] event, but only one operation wins the bus arbitration. To determine which operation caused the interrupt, software must verify that its transmit operation actually completed before assuming that the received data is the result of its read operation.

Problems could also arise if the MPC8555E's I²C controller master sets up a transmit buffer and BD for a write request, but then is the target of a read request from another master. Without software precautions, the I²C controller responds to the other master with the transmit buffer originally intended for its own write request. To avoid this situation, a higher-level handshake protocol must be used. For example, a master, before reading a slave, writes the slave with a description of the requested data (which register should be read, for example). This operation is typical with many I²C devices.

In addition, it is not recommended to enable the MPC8555E I²C controller while another I²C master is executing transactions on the bus. The MPC8555E I²C controller should wait for the bus to become idle.

The MPC8555E I²C controller assumes that other I²C devices on the bus closely conform to the I²C specification. Unexpected behavior can occur if the MPC8555E I²C controller is connected with devices which operate outside the specification. For example, a slave device which acknowledges a master write with 2 SCL pulses instead of 1 (total of 10 SCL pulses), can cause wrong behavior of the MPC8555E I²C controller on its next transaction.

44.4 I2C Registers

The following sections describe the I2C registers.

44.4.1 I2C Mode Register (I2MOD)

The I2C mode register, shown in Figure 44-6, controls the I2C modes and clock source.

	0	1	2	3	4	5	6	7
Field	—		REVD	GCD	FLT	PDIV		EN
Reset	0000_0000							
R/W	R/W							
Offset	0x0x9_1860							

Figure 44-6. I2C Mode Register (I2MOD)

Table 44-1 describes I2MOD bit functions.

Table 44-1. I2MOD Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	REVD	Reverse data. Determines the Rx and Tx character bit order. 0 Normal operation. The msb (bit 0) of a character is transferred first. 1 Reverse data. the lsb (bit 7) of a character is transferred first. **Note:** Clearing REVD is strongly recommended to ensure consistent bit ordering across devices.
3	GCD	General call disable. Determines whether the receiver acknowledges a general call address. 0 General call address is enabled. 1 General call address is disabled.
4	FLT	Clock filter. Determines if the I2C input clock SCL is filtered to prevent spikes in a noisy environment. 0 SCL is not filtered. 1 SCL is filtered by a digital filter.
5–6	PDIV	Predivider. Selects the clock division factor before it is input into the I2C BRG. The clock source for the I2C BRG is the BRGCLK generated from the CPM clock. 00 BRGCLK/32 01 BRGCLK/16 10 BRGCLK/8 11 BRGCLK/4 **Note:** To both save power and reduce noise susceptibility, select the PDIV with the largest division factor (slowest clock) that still meets performance requirements.
7	EN	Enable I2C operation 0 I2C is disabled. The I2C is in a reset state and consumes minimal power. 1 I2C is enabled. Do not change other I2MOD bits when EN is set.

44.4.2 I²C Address Register (I2ADD)

The I²C address register, shown in Figure 44-7, holds the address for this I²C port.

	0	6	7
Field	SAD		—
Reset	Undefined		
R/W	R/W		
Offset	0x0x9_1864		

Figure 44-7. I²C Address Register (I2ADD)

Table 44-2 describes I2ADD fields.

Table 44-2. I2ADD Field Descriptions

Bits	Name	Description
0–6	SAD	Slave address 0–6. Holds the slave address for the I²C port.
7	—	Reserved, should be cleared.

44.4.3 I²C Baud Rate Generator Register (I2BRG)

The I²C baud rate generator register, shown in Figure 44-8, sets the divide ratio of the I²C BRG.

	0	7
Field	DIV	
Reset	1111_1111	
R/W	R/W	
Offset	0x0x9_1868	

Figure 44-8. I²C Baud Rate Generator Register (I2BRG)

Table 44-3 describes I2BRG fields.

Table 44-3. I2BRG Field Descriptions

Bits	Name	Description
0–7	DIV	Division ratio 0–7. Specifies the divide ratio of the BRG divider in the I²C clock generator. The output of the prescaler is divided by 2 × ([DIV0–DIV7] + 3) and the clock has a 50% duty cycle. DIV must be programmed to a minimum value of 2 if the digital filter is disabled and 6 if it is enabled.

44.4.4 I²C Event/Mask Registers (I2CER/I2CMR)

The I²C event register (I2CER) is used to generate interrupts and report events. When an event is recognized, the I²C controller sets the corresponding I2CER bit. I2CER bits are cleared by writing ones; writing zeros has no effect. Setting a bit in the I²C mask register (I2CMR) enables and clearing a bit masks

the corresponding interrupt. Unmasked I2CER bits must be cleared before the CP clears internal interrupt requests. Figure 44-9 shows both registers.

	0	2	3	4	5	6	7
Field	—		TXE	—	BSY	TXB	RXB
Reset	0000_0000						
R/W	R/W						
Offset	0x0x9_1870(I2CER)/0x0x9_1874 I2CMR)						

Figure 44-9. I²C Event/Mask Registers (I2CER/I2CMR)

Table 44-4 describes the I2CER/I2CMR fields.

Table 44-4. I2CER/I2CMR Field Descriptions

Bits	Name	Description
0–2	—	Reserved and should be cleared.
3	TXE	Tx error. Set when an error occurs during transmission.
4	—	Reserved and should be cleared.
5	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.
6	TXB	Tx buffer. Set when the Tx data of the last character in the buffer is written to the Tx FIFO. Two character times must elapse to guarantee that all data has been sent.
7	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the RxBD is closed.

44.4.5 I²C Command Register (I2COM)

The I²C command register, shown in Figure 44-10, is used to start I²C transfers and to select master or slave mode.

	0	1	2	3	4	5	6	7
Field	STR	—						M/S
Reset	0000_0000							
R/W	R/W							
Offset	0x0x9_186C							

Figure 44-10. I²C Command Register (I2COM)

Table 44-5 describes I2COM fields.

Table 44-5. I2COM Field Descriptions

Bits	Name	Description
0	STR	Start transmit. In master mode, setting STR causes the I²C controller to start sending data from the I²C Tx buffers if they are ready. In slave mode, setting STR when the I²C controller is idle causes it to load the Tx data register from the I²C Tx buffer and start sending when it receives an address byte that matches the slave address with R/W = 1. STR is always read as a 0.
1–6	—	Reserved and should be cleared.
7	M/S	Master/slave. Configures the I²C controller to operate as a master or a slave. 0 I²C is a slave. 1 I²C is a master.

44.5 I²C Parameter RAM

The I²C controller parameter table is used for the general I²C parameters and is similar to the SCC general-purpose parameter RAM. The CP accesses the I²C parameter table using a user-programmed pointer (I2C_BASE) located in the parameter RAM; see Section 21.4.2, "Parameter RAM." The I²C parameter table can be placed at any 64-byte aligned address in the dual-port RAM's general-purpose area (banks 1–8, 11, and 12). The user must initialize certain parameter RAM values before the I²C is enabled; the CP initializes the other values. Software usually does not access parameter RAM entries once they are initialized; they should be changed only when the I²C is inactive.

Table 44-6 shows the I²C parameter memory map.

Table 44-6. I²C Parameter RAM Memory Map

Offset [1]	Name[2]	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address. Indicate where the BD tables begin in the dual-port RAM. Setting Rx/TxBD[W] in the last BD in each BD table determines how many BDs are allocated for the Tx and Rx sections of the I²C. Initialize RBASE/TBASE before enabling the I²C. Furthermore, do not configure BD tables of the I²C to overlap any other active controller's parameter RAM. RBASE and TBASE should be divisible by eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code registers. The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. See Figure 44-11 and Table 44-7.
0x05	**TFCR**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. Defines the maximum number of bytes the MPC8555E writes to a Rx buffer before moving to the next buffer. The MPC8555E writes fewer bytes to the buffer than the MRBLR value if an error or end-of-frame occurs. Buffers should not be smaller than MRBLR. Tx buffers are unaffected by MRBLR and can vary in length; the number of bytes to be sent is specified in TxBD[Data Length]. MRBLR is not intended to be changed while the I²C is operating. However it can be changed in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). The change takes effect when the CP moves control to the next RxBD. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the I²C receiver is disabled. MRBLR should be greater than zero; it should be an even number if the character length of the data exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state.[3] Reserved for CP use.

Table 44-6. I²C Parameter RAM Memory Map (continued)

Offset [1]	Name[2]	Width	Description
0x0C	RPTR	Word	Rx internal data pointer[3] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the next descriptor the receiver transfers data to when it is in an idle state or to the current descriptor during frame processing for each I²C channel. After a reset or when the end of the descriptor table is reached, the CP initializes RBPTR to the value in RBASE. Most applications should not write RBPTR, but it can be modified when the receiver is disabled or when no receive buffer is used.
0x12	RCOUNT	Hword	Rx internal byte count[3] is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	RTEMP	Word	Rx temp.[3] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state.[3] Reserved for CP use.
0x1C	TPTR	Word	Tx internal data pointer [3] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the next descriptor that the transmitter transfers data from when it is in an idle state or to the current descriptor during frame transmission. After a reset or when the end of the descriptor table is reached, the CP initializes TBPTR to the value in TBASE.Most applications should not write TBPTR, but it can be modified when the transmitter is disabled or when no transmit buffer is used.
0x22	TCOUNT	Hword	Tx internal byte count[3] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.
0x24	TTEMP	Word	Tx temp.[3] Reserved for CP use.
0x34	SDMATMP	Word	SDMA temp.[3] Reserved for CP use.

1 From the pointer value programmed in I2C_BASE at CCSRBAR + 0x8AFC.

2 **Boldfaced** entries must be initialized by the user.

3 Normally, these parameters need not be accessed.

Figure 44-11 shows the RFCR, TFCR bit fields.

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**TC2**	**DTB**	—
Reset	0000_0000							
R/W	R/W							
Offset	I2C_BASE + 04 (RFCR)/I2C_BASE + 05 (TFCR)							

Figure 44-11. I²C Function Code Registers (RFCR, TFCR)

Table 44-7 describes the RFCR/TFCR bit fields.

Table 44-7. RFCR, TFCR Field Descriptions

Bits	Name[1]	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global access bit 0 Disable memory snooping 0 Enable memory snooping
3–4	**BO**	Byte ordering. Selects the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame or BD. 00 True little-endian. Note this mode can only be used with 32-bit port size memory. 01 Munged little-endian. 1x Big-endian.
5	**TC2**	Transfer code 2. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0:1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Data bus indicator. 0 Use system bus for SDMA operation. 1 Reserved.
7	—	Reserved, should be cleared.

[1] **Boldfaced** entries must be initialized by the user.

44.6 I²C Commands

The I²C transmit and receive commands, shown in Table 44-8, are issued to the CP command register (CPCR).

Table 44-8. I²C Transmit/Receive Commands

Command	Description
INIT TX PARAMETERS	Initializes all transmit parameters in the parameter RAM to their reset state. Should be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.
CLOSE RXBD	Forces the I²C controller to close the current Rx BD and use the next BD for subsequently received data. If the controller is not receiving data, no action is taken. Use this command to extract data from a partially full buffer.
INIT RX PARAMETERS	Initializes all receive parameters in the parameter RAM to their reset state. Should be issued only when the receiver is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

44.7 I²C Buffer Descriptor (BD) Table

As shown in Figure 44-12, buffer descriptors (BDs) are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as for the SCCs and SMCs and form circular queues that determine the order buffers are transferred. The CP uses BDs to confirm reception and transmission or to indicate error conditions so that the core knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the dual-port RAM.

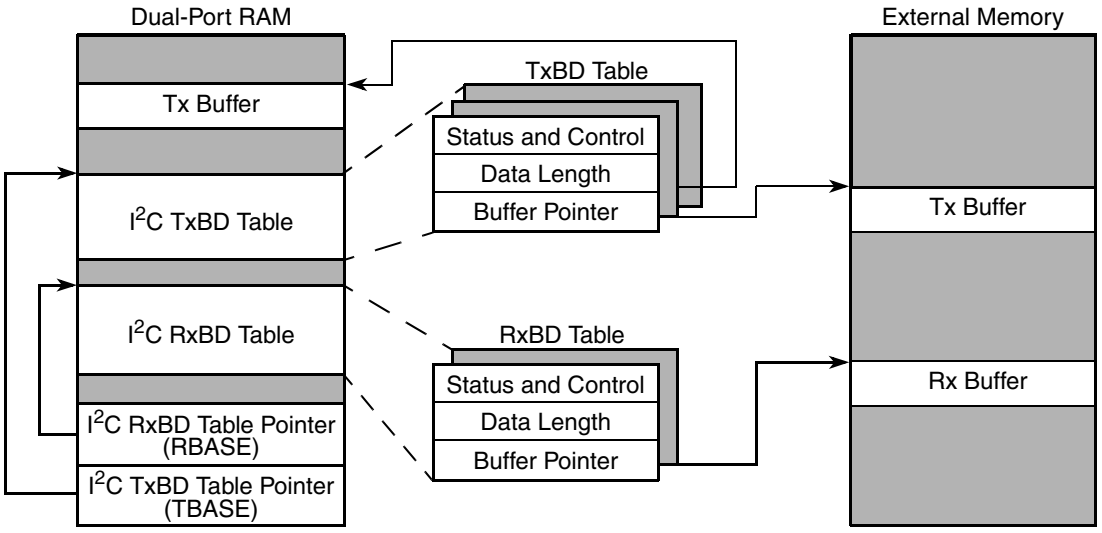


Figure 44-12. I²C Memory Structure

44.7.1 I²C Buffer Descriptors (BDs)

Receive and transmit buffer descriptors report information about each buffer transferred and whether a maskable interrupt should be generated. Each 64-bit BD, shown in Figure 44-13 and Figure 44-14, has the following structure:

- The half word at offset + 0 contains status and control bits. The CP updates the status bits after the buffer is sent or received.
- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
  - For an RxBD, this is the number of octets the CP writes into this RxBD's buffer once the descriptor closes. The CP updates this field after the received data is placed into the associated buffer. Memory allocated for this buffer should be no smaller than MRBLR.
  - For a TxBD, this is the number of octets the CP should transmit from its buffer. Normally, this value should be greater than zero. The CP never modifies this field.
- The word at offset + 4 points to the beginning of the buffer.
  - For an RxBD, the pointer must be even and can point to internal or external memory.
  - For a TxBD, the pointer can be even or odd. The buffer can reside in internal or external memory.

44.7.1.1 I²C Receive Buffer Descriptor (RxBD)

Using RxBDs, the CP reports on each buffer received, closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer when the current one is full. It closes the buffer when a stop or start condition is found on the I²C bus or when an overrun error occurs. The core should write RxBD bits before the I²C controller is enabled.

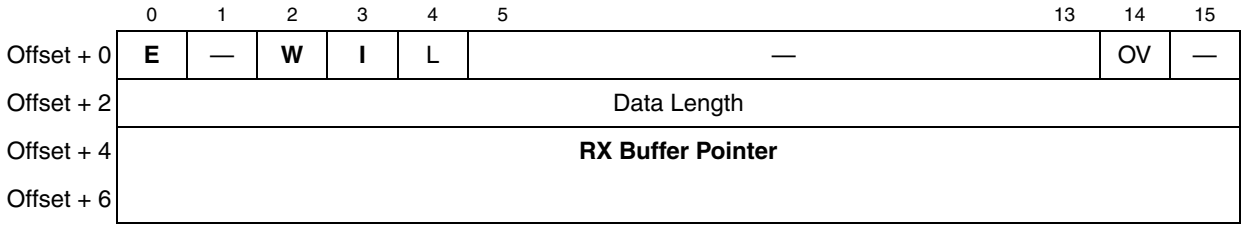


Figure 44-13. I²C RxBD

Table 44-9 describes I²C RxBD status and control bits.

Table 44-9. I²C RxBD Status and Control Bits

Bits	Name[1]	Description
0	**E**	Empty 0 The buffer is full or stopped receiving because of an error. The core can examine or write to any fields of this RxBD, but the CP does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved and should be cleared.
2	**W**	Wrap (last BD in table) 0 Not the last BD in the RxBD table 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is full. 1 The I2CER[RXB] is set when the CP fills this buffer, indicating that the core needs to process the buffer. The RXB bit can cause an interrupt if it is enabled.
4	L	Last. The I²C controller sets L. 0 This buffer does not contain the last character of the message. 1 This buffer holds the last character of the message. The I²C controller sets L after all received data is placed into the associated buffer, or because of a stop or start condition or an overrun.
5–13	—	Reserved and should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception. The I²C controller updates this bit after the received data is placed into the associated buffer.
15	—	Reserved and should be cleared.

[1] **Boldfaced** entries must be initialized by the user.

44.7.1.2 I²C Transmit Buffer Descriptor (TxBD)

Transmit data is arranged in buffers referenced by TxBDs in the TxBD table. The first word of the TxBD, shown in Figure 44-14, contains status and control bits.

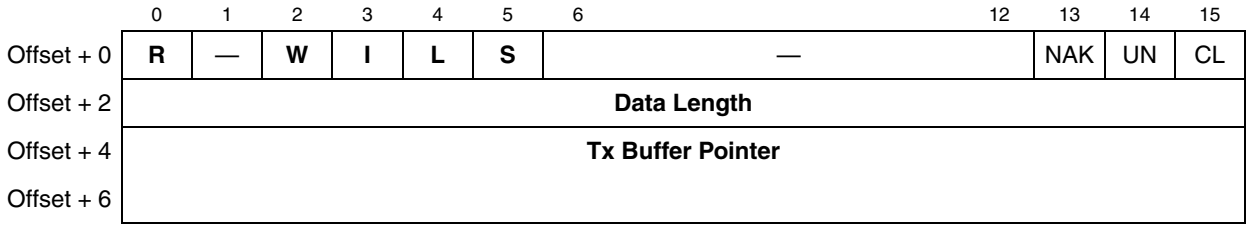


Figure 44-14. I²C TxBD

Table 44-10 describes I²C TxBD status and control bits.

Table 44-10. I²C TxBD Status and Control Bits

Bits	Name[1]	Description
0	**R**	Ready. 0 The buffer is not ready to be sent. This BD or its buffer can be modified. The CP clears R after the buffer is sent or an error occurs. 1 The buffer is ready for transmission or is being sent. The BD cannot be modified once R is set.
1	—	Reserved and should be cleared.
2	**W**	Wrap (last BD in TxBD table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP transmits data using the BD pointed to by TBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is serviced. 1 I2CER[TXB] or I2CER[TXE] is set when the buffer is serviced. If enabled, an interrupt occurs.
4	**L**	Last. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message. The I²C controller generates a stop condition after sending this buffer.
5	**S**	Generate start condition. Provides ability to send back-to-back frames with one I2COM[STR] trigger. 0 Do not send a start condition before the first byte of the buffer. 1 Send a start condition before the first byte of the buffer. (Used to separate frames.) Note: If this BD is the first one in the frame when I2COM[STR] is triggered, a start condition is sent regardless of the value of TxBD[S].
6–12	—	Reserved and should be cleared.
13	NAK	No acknowledge. Indicates that the transmission was aborted because the last byte sent was not acknowledged. The I²C controller updates NAK after the buffer is sent.
14	UN	Underrun. Indicates that the I²C controller encountered a transmitter underrun condition while sending the associated buffer. The I²C controller updates UN after the buffer is sent.
15	CL	Collision. Indicates that transmission terminated because the transmitter was lost while arbitrating for the bus. The I²C controller updates CL after the buffer is sent.

[1] **Boldfaced** entries must be initialized by the user.

Chapter 45
Parallel I/O Ports

The CPM supports four general-purpose I/O ports—ports A, B, C, and D. Each pin in the I/O ports can be configured as a general-purpose I/O signal or as a dedicated peripheral interface signal. Port C is unique in that 16 of its pins (PC[0:1,4:15,23,29]) can generate interrupts to the interrupt controller.

Each pin can be configured as an input or output and has a latch for data output, read or written at any time, and configured as general-purpose I/O or a dedicated peripheral pin. Part of the pins 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 internal pull-up resistors. Due to the CPM'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 applications. Note that to obtain a full understanding of the pin assignment capability described in this chapter, a user must understand the CPM peripherals.

45.1 Features

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

- Port A is 24 bits.
- Port B is 14 bits.
- Port C is 28 bits.
- Port D is 16 bits.
- All ports are bidirectional.
- All ports have alternate on-chip peripheral functions.
- All ports are three-stated at system reset.
- All pin values can be read while the pin is connected to an on-chip peripheral.
- Some pins have open-drain capability.
- Port C offers 16 interrupt input pins (PC[0:1, 4:15, 23, 29]).

45.2 Port Registers

Each port has four memory-mapped, read/write, 32-bit control registers.

45.2.1 Port Open-Drain Registers (PODR*x*)

The port open-drain registers (PODR*x*), shown in Figure 45-1, Figure 45-2, Figure 45-3 and Figure 45-4, indicate a normal or wired-OR configuration of the port pins.

45.2.1.1 Port A Open-Drain Register (PODRA)

	0–7	8	9	10	11	12	13	14	15
Field	—	OD8	OD9	OD10	OD11	OD12	OD13	OD14	OD15
Reset	0000_0000_0000_0000								
R/W	R/W								
Offset	0x9_0D0C								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	OD30	OD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D0E															

Figure 45-1. Port A Open-Drain Registers (PODRA)

Table 45-1 describes PODRA fields.

Table 45-1. PODRA Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–31	OD*n*	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.

45.2.1.2 Port B Open-Drain Register (PODRB)

	0–15
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Offset	0x9_0D2C

	16–17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	OD30	OD31
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_0D2E														

Figure 45-2. Port B Open-Drain Registers (PODRB)

Table 45-2 describes PODRB fields.

Table 45-2. PODRB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared.
18–31	OD*n*	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.

45.2.1.3 Port C Open-Drain Register (PODRC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OD0	OD1	—		OD4	OD5	OD6	OD7	OD8	OD9	OD10	OD11	OD12	OD13	OD14	OD15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D4C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D4E															

Figure 45-3. Port C Open-Drain Registers (PODRC)

Table 45-3 describes PODRC fields.

Table 45-3. PODRC Field Descriptions

Bits	Name	Description
0–1, 4–29	OD*n*	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.
2–3, 30–31	—	Reserved, should be cleared.

45.2.1.4 Port D Open-Drain Register (PODRD)

	0	6	7	8	13	14	15
Field	—		OD7	—		OD14	OD15
Reset	0000_0000_0000_0000						
R/W	R/W						
Offset	0x9_0D0C (PODRA), 0x9_0D2C (PODRB), 0x9_0D4C (PODRC), 0x9_0D6C (PODRD)						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	—		OD29	OD30	OD31
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_0D0E (PODRA), 0x9_0D2E (PODRB), 0x9_0D4E (PODRC), 0x9_0D6E (PODRD)														

Figure 45-4. Port D Open-Drain Registers (PODRD)

Table 45-4 describes PODRD fields.

Table 45-4. PODRD Field Descriptions

Bits	Name	Description
0–6, 8–13, 26–28	—	Reserved, should be cleared.
7, 14–25, 29–31	OD*n*	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.

45.2.2 Port Data Registers (PDAT*x*)

A read of port data registers (PDAT*x*), shown in Figure 45-5, Table 45-6, Table 45-7, and Table 45-8, 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 PDAT*x* is latched, and if the equivalent PDIR*x* bit is configured as an output, the value latched for that bit is driven onto its respective pin. PDAT*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 port data register (PDAT*x*). Data written to the PDAT*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 PDAT*x* is read, the port pin itself is read. If a port pin is configured as an input, data written to PDAT*x* is still stored in the output latch, but is prevented from reaching the port pin. In this case, when PDAT*x* is read, the state of the port pin is read.

45.2.2.1 Port A Data Register (PDATA)

	0	7	8	9	10	11	12	13	14	15
Field	—		D8	D9	D10	D11	D12	D13	D14	D15
Reset	—									
R/W	R/W									
Offset	0x9_0D10									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	—															
R/W	R/W															
Offset	0x9_0D12															

Figure 45-5. Port A Data Registers (PDATA)

45.2.2.2 Port B Data Register (PDATB)

	0	15
Field	—	
Reset	—	
R/W	R/W	
Offset	0x9_0D30	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	—															
R/W	R/W															
Offset	0x9_0D32															

Figure 45-6. Port B Data Registers (PDATB)

45.2.2.3 Port C Data Register (PDATC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	D0	D1	—		D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	—															
R/W	R/W															
Offset	0x9_0D50															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	—	
Reset	—															
R/W	R/W															
Offset	0x9_0D52															

Figure 45-7. Port C Data Registers (PDATC)

45.2.2.4 Port D Data Register (PDATD)

	0						6	7	8					13	14	15
Field	—							D7	—						D14	D15
Reset	—															
R/W	R/W															
Offset	0x9_0D70															

	16	17	18	19	20	21	22	23	24	25	26		28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	—			D29	D30	D31
Reset	—															
R/W	R/W															
Offset	0x9_0D72															

Figure 45-8. Port D Data Registers (PDATD)

45.2.3 Port Data Direction Registers (PDIR*x*)

The port data direction registers (PDIR*x*), shown in Figure 45-9, Figure 45-10, Figure 45-11, and Figure 45-12, are cleared at system reset.

45.2.3.1 Port A Data Direction Register (PDIRA)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—								DR8	DR9	DR10	DR11	DR12	DR13	DR14	DR15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D00															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DR16	DR17	DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	DR30	DR31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D02															

Figure 45-9. Port A Data Direction Register (PDIRA)

Table 45-5 describes PDIRA fields.

Table 45-5. PDIRA Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–31	DR*n*	Direction. Indicates whether the pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

45.2.3.2 Port B Data Direction Register (PDIRB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Offset	0x9_0D20	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	DR30	DR31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D22															

Figure 45-10. Port B Data Direction Register (PDIRB)

Table 45-6 describes PDIRB fields.

Table 45-6. PDIRB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared.
18–31	DR*n*	Direction. Indicates whether the pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

45.2.3.3 Port C Data Direction Register (PDIRC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DR0	DR1	—		DR4	DR5	DR6	DR7	DR8	DR9	DR10	DR11	DR12	DR13	DR14	DR15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D40															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DR16	DR17	DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D42															

Figure 45-11. Port C Data Direction Register (PDIRC)

Table 45-7 describes PDIRC fields.

Table 45-7. PDIRC Field Descriptions

Bits	Name	Description
0–1, 4–29	DR*n*	Direction. Indicates whether the pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.
2–3, 30–31	—	Reserved, should be cleared.

45.2.3.4 Port D Data Direction Register (PDIRD)

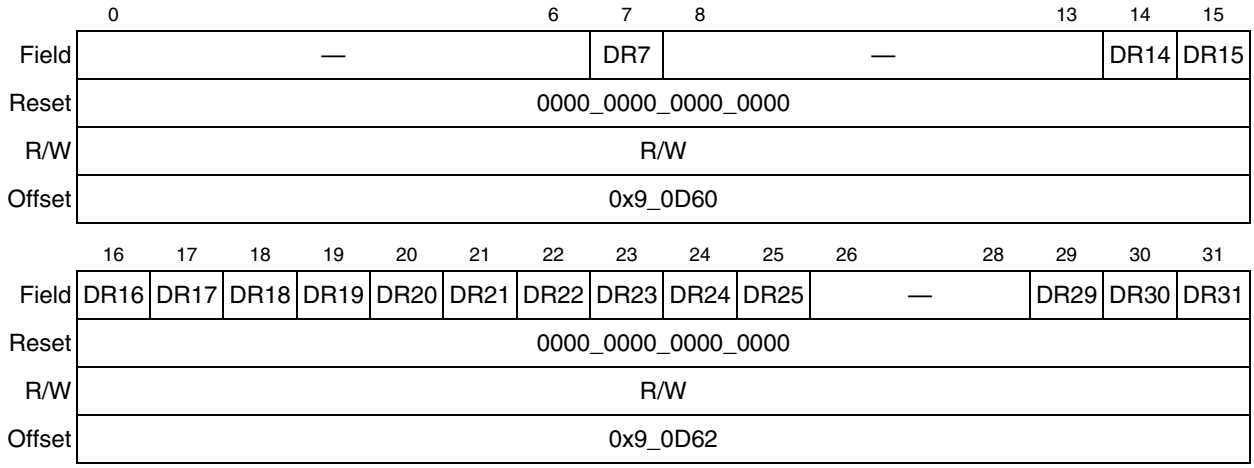


Figure 45-12. Port D Data Direction Register (PDIRD)

Table 45-8 describes PDIRD fields.

Table 45-8. PDIRD Field Descriptions

Bits	Name	Description
0–6, 8–13, 26–28	—	Reserved, should be cleared.
7, 14–25, 29–31	DR*7*	Direction. Indicates whether the pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

45.2.4 Port Pin Assignment Registers (PPAR*x*)

The port pin assignment registers (PPAR*x*) are cleared at system reset.

45.2.4.1 Port A Pin Assignment Registers (PPARA)

	0	7	8	9	10	11	12	13	14	15
Field	—		DD8	DD9	DD10	DD11	DD12	DD13	DD14	DD15
Reset	0000_0000_0000_0000									
R/W	R/W									
Offset	0x9_0D04									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	DD26	DD27	DD28	DD29	DD30	DD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D06															

.

Figure 45-13. Port A Pin Assignment Register (PPARA)

Table 45-9 describes PPARA fields.

Table 45-9. PPARA Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–21	DD*n*	Dedicated enable. Indicates whether the pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRA.
22–23	DD*n*	Dedicated enable. Refer to description above.
	—	Reserved, should be cleared. **Note:** PA[22:23] can be used only as general purpose.
24–31	DD*n*	Dedicated enable. Indicates whether the pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRA.

45.2.4.2 Port B Pin Assignment Registers (PPARB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Offset	0x9_0D24	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	DD26	DD27	DD28	DD29	DD30	DD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D26															

Figure 45-14. Port B Pin Assignment Register (PPARB)

Table 45-10 describes PPARB fields.

Table 45-10. PPARB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared.
18–31	DD*n*	Dedicated enable. Indicates whether the pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRB.

45.2.4.3 Port C Pin Assignment Registers (PPARC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DD0	DD1	—		DD4	DD5	DD6	DD7	DD8	DD9	DD10	DD11	DD12	DD13	DD14	DD15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D44															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	DD26	DD27	DD28	DD29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D46															

Figure 45-15. Port C Pin Assignment Register (PPARC)

Table 45-11 describes PPARC fields.

Table 45-11. PPARC Field Descriptions

Bits	Name	Description
0–1, 4-29	DD*n*	Dedicated enable. Indicates whether the pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRC.
2–3, 30–31	—	Reserved, should be cleared.

45.2.4.4 Port D Pin Assignment Registers (PPARD)

	0	6	7	8	13	14	15
Field	—		DD7	—		DD14	DD15
Reset	0000_0000_0000_0000						
R/W	R/W						
Offset	0x9_0D64						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	—		DD29	DD30	DD31
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_0D66														

Figure 45-16. Port D Pin Assignment Register (PPARD)

Table 45-12 describes PPARD fields.

Table 45-12. PPARD Field Descriptions

Bits	Name	Description
0–6, 8–13, 26–28	–	Reserved, should be cleared.
7, 14–25, 29–31	DD*n*	Dedicated enable. Indicates whether the pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRD.

45.2.5 Port Special Options Registers (PSOR*x*)

PSOR*x* bits are effective only if the corresponding PPAR*x*[DD*n*] = 1 (a dedicated peripheral function).

NOTE

If the corresponding PPAR*x*[DD*n*] = 1 (configured as a general-purpose pin) before programming a PSOR*x* or PDIR*x* bit, a pin might function for a short period as an unwanted dedicated function and cause unknown behavior.

Figure 45-17 shows the port special options registers (PSOR*x*).

45.2.5.1 Port A Special Options Register (PSORA)

	0–7	8	9	10	11	12	13	14	15
Field	—	SO8	SO9	SO10	SO11	SO12	SO13	SO14	SO15
Reset	0000_0000_0000_0000								
R/W	R/W								
Offset	0x9_0D08								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	SO26	SO27	SO28	SO29	SO30	SO31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D0A															

Figure 45-17. Port A Special Options Register (PSORA)

Table 45-13 describes PSORA fields.

Table 45-13. PSOR*x* Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–21	SO*n*	Special-option. Determines whether a pin configured for a dedicated function (PPARA[DD*n*] = 1) uses option 1 or option 2. Options are described in Section 45.5, “Port Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
22–23	SO*n*	Special-option. See description above.
	—	Reserved, should be cleared. **Note:** PA[22:23] can be used only as general purpose.
24–31	SO*n*	Special-option. Determines whether a pin configured for a dedicated function (PPARA[DD*n*] = 1) uses option 1 or option 2. Options are described in Section 45.5, “Port Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

45.2.5.2 Port B Special Options Registers (PSORB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Offset	0x9_0D28	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	SO26	SO27	SO28	SO29	SO30	SO31
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D2A															

Figure 45-18. Port B Special Options Registers (PSORB)

Table 45-14 describes PSORB fields.

Table 45-14. PSORB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared.
18–31	SO*n*	Special-option. Determines whether a pin configured for a dedicated function (PPARB[DD*n*] = 1) uses option 1 or option 2. Options are described in Section 45.5, “Port Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

45.2.5.3 Port C Special Options Registers (PSORC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SO0	SO1	—		SO4	SO5	SO6	SO7	SO8	SO9	SO10	SO11	SO12	SO13	SO14	SO15
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D48															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	SO26	SO27	SO28	SO29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Offset	0x9_0D4A															

Figure 45-19. Port C Special Options Registers (PSORC)

Table 45-15 describes PSORC fields.

Table 45-15. PSORC Field Descriptions

Bits	Name	Description
0–1, 4–29	SO*n*	Special-option. Determines whether a pin configured for a dedicated function (PPAR*C*[DD*n*] = 1) uses option 1 or option 2. Options are described in Section 45.5, “Port Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
2–3 30–31	—	Reserved, should be cleared.

45.2.5.4 Port D Special Options Registers (PSORD)

	0						6	7	8				13	14	15
Field	—							SO7	–					SO14	SO15
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_0D68														

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	–		SO29	SO30	SO31
Reset	0000_0000_0000_0000														
R/W	R/W														
Offset	0x9_0D6A														

Figure 45-20. Special Options Registers (PSORD)

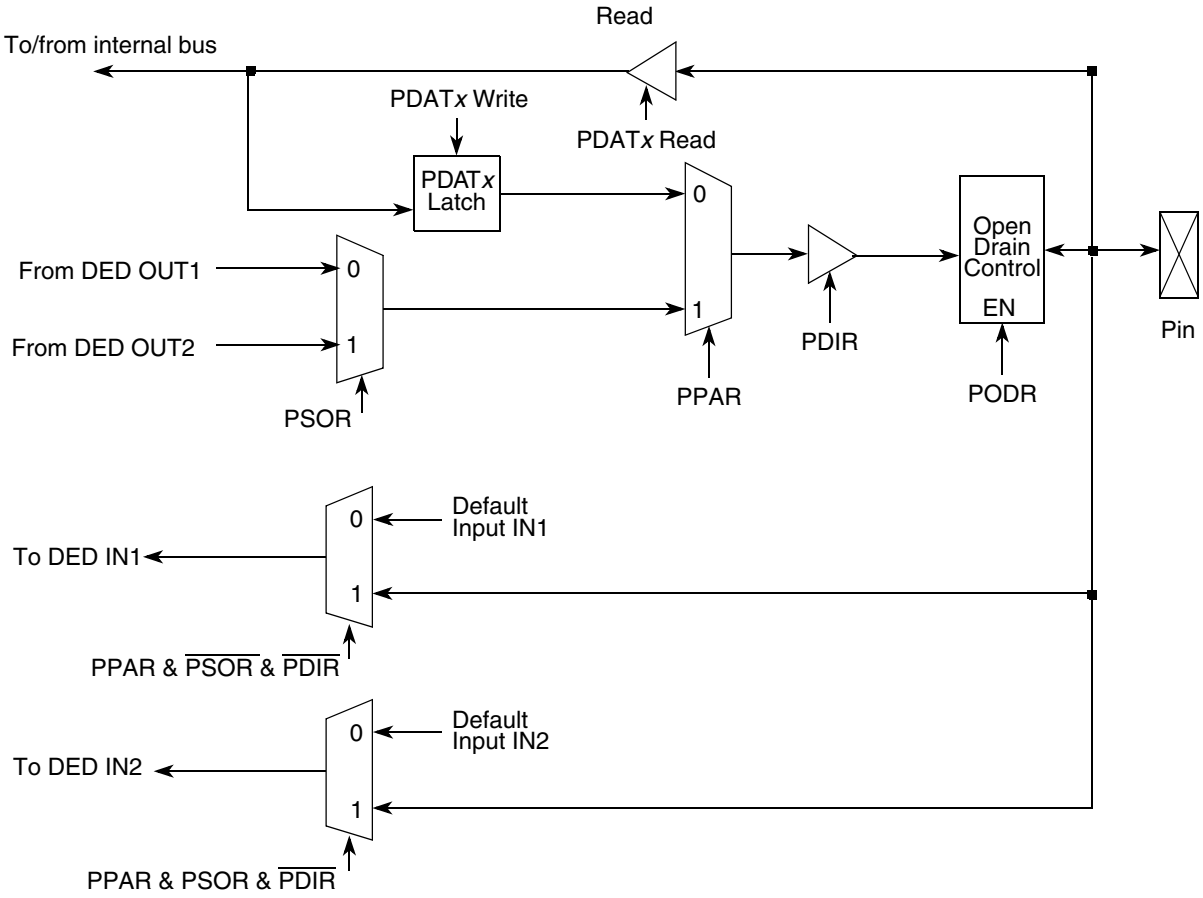
Table 45-13 describes PSORD fields.

Table 45-16. PSORD Field Descriptions

Bits	Name	Description
0–6, 8–13, 26–28	—	Reserved, should be cleared.
7, 14–25, 29–31	SO*n*	Special-option. Determines whether a pin configured for a dedicated function (PPARD[DD*n*] = 1) uses option 1 or option 2. Options are described in Section 45.5, “Port Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

45.3 Port Block Diagram

Figure 45-21 shows the functional block diagram.



Register Name	0	1	Description
PPAR*x*	General purpose	Dedicated	Port pin assignment
PSOR*x*	Dedicated 1	Dedicated 2	Special operation
PDIR*x*	Input	Output	Direction[1]
PODR*x*	Regular	Open drain	
PDAT*x*	0	1	Data

[1] Bidirectional signals must be programmed as inputs (PDIR = 0).

Figure 45-21. Port Functional Operation

45.4 Port Pin Functions

Each pin can operate as a general-purpose I/O pin or as a dedicated input or output pin.

45.4.1 General-Purpose I/O Pins

Each one of the port pins is independently configured as a general-purpose I/O pin if the corresponding port pin assignment register (PPAR) bit is cleared. Each pin is configured as a dedicated on-chip peripheral pin if the corresponding PPAR bit is set.When the port pin is configured as a general-purpose I/O pin, the signal direction for that pin is determined by the corresponding control bit in the port data direction register (PDIR). The port I/O pin is configured as an input if the corresponding PDIR bit is cleared; it is configured as an output if the corresponding PDIR bit is set. All PPAR and PDIR bits are cleared on total system reset, configuring all port pins as general-purpose input pins.

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

45.4.2 Dedicated Pins

When a port pin 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 right-most column.

NOTE

Some output functions can be output on 2 different pins. For example, the output for BRG5 can come out on both PC13 and PC23. The user can freely configure such functions to be output on two pins 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 two pins.

Many input functions can also come from two different pins; see Section 45.5, “Port Tables.”

45.5 Port Tables

Tables 45-17 through 45-20 describe the port functionality according to the configuration of the port registers (PPAR*x*, PSOR*x*, and PDIR*x*). Each pin can function as a general-purpose I/O, one of two dedicated outputs, or one of two dedicated inputs.

As shown in Figure 45-22, some input functions can come from two different pins for flexibility. Secondary option programming is relevant only if primary option is programmed to the default value.

NOTE

In the MPC8555E CPM PIO port pinmuxing, FCC2 only has a primary option available for FCC2_TxAddr0, FCC2_RxAddr0. There are no secondary options for these signals in the CPM PIO port pinmuxing.

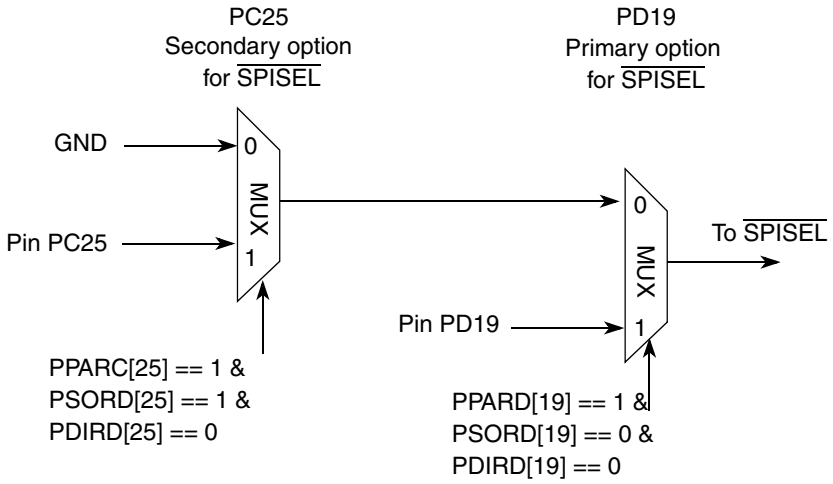


Figure 45-22. Primary and Secondary Option Programming

In the tables below, 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.

Table 45-17 shows the port A pin assignments.

Table 45-17. Port A Dedicated Pin Assignment (PPARA = 1)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or I/O if Specified)	Default Input
PA31	**FCC1: TxEnb** UTOPIA master	**FCC1: TxEnb** UTOPIA slave	GND		**FCC1: COL** MII	GND
PA30	**FCC1: TxClav** UTOPIA slave	**FCC1: TxClav** UTOPIA master **FCC1: TxClav0** MPHY, master, direct polling	GND	**FCC1: $\overline{\text{RTS}}$**	**FCC1: CRS** MII	GND
PA29	**FCC1: TxSOC** UTOPIA			**FCC1: TX_ER** MII		
PA28	**FCC1: RxEnb** UTOPIA master	**FCC1: RxEnb** UTOPIA slave	GND	**FCC1: TX_EN** MII/RMII		
PA27		**FCC1: RxSOC** UTOPIA	GND		**FCC1: RX_DV** MII **FCC1: CRS_DV** RMII	GND

Table 45-17. Port A Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or I/O if Specified)	Default Input
PA26	**FCC1: RxClav** UTOPIA slave	**FCC1: RxClav** UTOPIA master **FCC1: RxClav0** MPHY, master, direct polling	GND		**FCC1: RX_ER** MII/RMII	GND
PA25	**FCC1: TxD[0]** UTOPIA 8			MSNUM[2][1]		
PA24	**FCC1: TxD[1]** UTOPIA 8			MSNUM[3] [1]		
PA23	**FCC1: TxD[2]** UTOPIA 8					
PA22	**FCC1: TxD[3]** UTOPIA 8					
PA21	**FCC1: TxD[4]** UTOPIA 8 **FCC1: TxD[3]** MII/HDLC nibble					
PA20	**FCC1: TxD[5]** UTOPIA 8 **FCC1: TxD[2]** MII/HDLC nibble					
PA19	**FCC1: TxD[6]** UTOPIA 8 **FCC1: TxD[1]** MII/HDLC nibble **FCC1: TxD[1]** **RMII dibit**					
PA18	**FCC1: TxD[7]** UTOPIA 8 **FCC1: TxD[0]** MII/HDLC nibble **FCC1: TxD[0]** RMII dibit **FCC1: TxD** HDLC/transp					

Table 45-17. Port A Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or I/O if Specified)	Default Input
PA17		**FCC1: RxD[7]** UTOPIA 8 **FCC1: RxD[0]** MII/HDLC nibble **FCC1: RxD[0]** **RMII dibit** **FCC1: RxD[0]** HDLC/transp.	GND			
PA16		**FCC1: RxD[6]** UTOPIA 8 **FCC1: RxD[1]** MII/HDLC nibble **FCC1: RxD[1]** **RMII dibit**	GND			
PA15		**FCC1: RxD[5]** UTOPIA 8 **FCC1: RxD[2]** MII/HDLC nibble	GND			
PA14		**FCC1: RxD[4]** UTOPIA 8 **FCC1: RxD[3]** MII/HDLC nibble	GND			
PA13		**FCC1: RxD[3]** UTOPIA 8	GND	MSNUM[4] [1]		
PA12		**FCC1: RxD[2]** UTOPIA 8	GND	MSNUM[5] [1]		
PA11		**FCC1: RxD[1]** UTOPIA 8	GND	MSNUM[6] [1]		
PA10		**FCC1: RxD[0]** UTOPIA 8	GND	MSNUM[7] [1]		
PA9	**SMC2: SMTXD**				**TDM_C2: L1TXD** I/O	
PA8		**SMC2: SMRXD** (primary option)	GND		**TDM_C2: L1RXD** I/O, serial	

[1] MSNUM[2–7] is the sub-block code of the peripheral controller using SDMA; MSNUM[7] indicates which section, transmit or receive, is active during the transfer. MSNUM is a field in the SMEVR and LMEVR registers that represents the serial number of the channel that created a data error. See Section 27.1.2, "SDMA Event Registers (SMEVR, LMEVR)," for more information.

Table 45-18 shows the port B pin assignments.

Table 45-18. Port B Dedicated Pin Assignment (PPARB = 1)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or I/O if Specified)	Default Input
PB31	**FCC2: TX_ER** MII	**FCC2: RxSOC** UTOPIA				GND
PB30	**FCC2: TxSOC** UTOPIA	**FCC2: RX_DV** MII **FCC2: CRS_DV** RMII	GND			GND
PB29	**FCC2: RxClav** UTOPIA slave	**FCC2: RxClav** UTOPIA master		**FCC2: TX_EN** MII/RMII		GND
PB28	**FCC2: $\overline{\text{RTS}}$**	**FCC2: RX_ER** MII/RMII	GND	**SCC1: TXD**	**TDM_C2: L1TSYNC/GRANT**	GND
PB27	**FCC2: TxD[0]** UTOPIA 8	**FCC2: COL** MII	GND		**TDM_B2: L1TXD I/O**	GND
PB26	**FCC2: TxD[1]** UTOPIA 8	**FCC2: CRS** MII	GND		**TDM_B2: L1RXD I/O**	GND
PB25	**FCC2: TxD[4]** UTOPIA 8 **FCC2: TxD[3]** MII/HDLC nibble				**TDM_B2: L1TSYNC/GRANT**	GND
PB24	**FCC2: TxD[5]** UTOPIA 8 **FCC2: TxD[2]** MII/HDLC nibble				**TDM_B2: L1RSYNC**	GND
PB23	**FCC2: TxD[6]** UTOPIA **FCC2: TxD[1]** MII/HDLC nibble **FCC2: TxD[1]** RMII dibit					GND
PB22	**FCC2: TxD[7]** UTOPIA **FCC2: TxD[0]** MII/HDLC nibble **FCC2: TxD[0]** RMII dibit **FCC2: TxD** HDLC/transp. serial					GND

Table 45-18. Port B Dedicated Pin Assignment (PPARB = 1) (continued)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or I/O if Specified)	Default Input
PB21		**FCC2: RxD[7]** UTOPIA 8 **FCC2: RxD[0]** MII/HDLC nibble **FCC2: RxD[0]** RMII dibit **FCC2: RxD** HDLC/transp. serial	GND			GND
PB20		**FCC2: RxD[6]** UTOPIA 8 **FCC2: RxD[1]** MII/HDLC nibble **FCC2: RxD[1]** RMII dibit	GND			GND
PB19		**FCC2: RxD[5]** UTOPIA 8 **FCC2: RxD[2]** MII/HDLC nibble	GND	**TDM_B2: $\overline{\text{L1RQ}}$**		
PB18		**FCC2: RxD[4]** UTOPIA 8 **FCC2: RxD[3]** MII/HDLC nibble	GND	**TDM_B2: L1CLKO**		

Table 45-19 shows the port C pin assignments.

Table 45-19. Port C Dedicated Pin Assignment (PPARC = 1)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or I/O if Specified)	Default Input
PC29	**BRG2: BRGO**	CLK3/TIN2	CLK7	**FCC2: TxAddr[4]** MPHY, master	**SCC1: $\overline{\text{CTS}}$**	GND
PC28	**Timer2: $\overline{\text{TOUT}}$**	CLK4/TIN1	CLK8		**SPI: SPICLK1** I/O (secondary option)	GND
PC27	**Timer1: $\overline{\text{TOUT}}$**	CLK5	GND	**BRG3: BRGO**	**FCC1: RxPrty** UTOPIA (secondary option)	GND

Table 45-19. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or I/O if Specified)	Default Input
PC26	**Timer3: $\overline{\text{TOUT}}$**	CLK6	GND	**FCC2: RxAddr[4]** MPHY, master		BRGO1
PC25		CLK7/TIN4	GND	**BRG4: BRGO**	**SPI: $\overline{\text{SPISEL}}$1** (secondary option)	VDD
PC24	**BRG1: BRGO**	CLK8/TIN3	GND	**Timer4: $\overline{\text{TOUT}}$**		GND
PC23	**BRG5: BRGO**	CLK9	CLK13	**FCC2: TxAddr[3]** MPHY, master	**SCC1: $\overline{\text{CD}}$**	GND
PC22	**FCC1: TxPrty** UTOPIA	CLK10	CLK14			
PC21	**BRG6: BRGO**	CLK11	CLK15			GND
PC20	**USB: $\overline{\text{OE}}$**	CLK12	CLK16	**FCC2: RxAddr[3]** MPHY, master		
PC19	**BRG7: BRGO**	CLK13	GND	**FCC2: TxAddr[2]** MPHY, master (secondary option)	**timer1/2: $\overline{\text{TGATE1}}$**	GND
PC18		CLK14	GND		**timer3/4: $\overline{\text{TGATE2}}$**	GND
PC17	**BRG8: BRGO**	CLK15	GND	**FCC2: TxAddr[0]** MPHY, master	**FCC2: TxAddr[0]** MPHY, slave	VDD
PC16		CLK16	GND	**FCC2: RxAddr[0]** MPHY, master	**FCC2: RxAddr[0]** MPHY, slave	
PC15		**SCC1: $\overline{\text{CTS}}$**	by PC29	**FCC1: TxAddr[0]** MPHY, master	**FCC1: TxAddr[0]** MPHY, slave	GND
PC14		**SCC1: $\overline{\text{CD}}$**	by PC23	**FCC1: RxAddr[0]** MPHY, master	**FCC1: RxAddr[0]** MPHY, slave	GND
PC13	**BRG5: BRGO**			**FCC1: TxAddr[1]** MPHY, master	**FCC1: TxAddr[1]** MPHY, slave	GND
PC12			VDD	**FCC1: RxAddr[1]** MPHY, master	**FCC1: RxAddr[1]** MPHY, slave	GND
PC11		**SCC3: $\overline{\text{CTS}}$** **USB: RP** (primary option)	by PC8	**FCC2: TxAddr[1]** MPHY, master	**FCC2: TxAddr[1]** MPHY, slave	
PC10		**SCC3: $\overline{\text{CD}}$** t **USB: RN**	GND	**FCC2: RxAddr[2]** MPHY, master		

Table 45-19. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or I/O if Specified)	Default Input
PC9		**SCC4: $\overline{\text{CTS}}$**	GND	**FCC2: TxAddr[2]** MPHY, master (primary option)	**TDM_A2: L1TSYNC/GRANT**	GND
PC8	**SCC1: $\overline{\text{RTS}}$**	**SCC4: $\overline{\text{CD}}$**	GND	**SI2: L1ST1** Strobe	**SCC3: $\overline{\text{CTS}}$1**	GND
PC7		**FCC1: $\overline{\text{CTS}}$**	GND	**FCC1: TxAddr[2]** MPHY master, multiplexed: polling	**FCC1: TxAddr[2]** MPHY, slave, multiplexed polling **FCC1: TxClav1** MPHY, master, direct polling	GND
PC6	**SI2: L1ST2** Strobe	**FCC1: $\overline{\text{CD}}$**	GND	**FCC1: RxAddr[2]** MPHY, master, multiplexed polling	**FCC1: RxAddr[2]** MPHY, slave, multiplexed polling) **FCC1: RxClav1** MPHY, master, direct polling	GND
PC5	**SMC1: SMTXD**			**SI2: L1ST3** Strobe	**FCC2: $\overline{\text{CTS}}$**	GND
PC4	**FCC2: TxPrty** UTOPIA	**SMC1: SMRXD**	GND	**SI2: L1ST4** Strobe	**FCC2: $\overline{\text{CD}}$**	GND
PC1	**BRG6: BRGO**	**FCC2: RxPrty** UTOPIA	GND	**TDM_A2: $\overline{\text{L1RQ}}$**	**TDM_C2: L1RSYNC**	GND
PC0	**BRG7: BRGO**		GND	**TDM_A2: L1CLKO**	**SMC1: SMSYN**	GND

Table 45-20 shows the port D pin assignments.

Table 45-20. Port D Dedicated Pin Assignment (PPARD = 1)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or I/O if Specified)	Default Input
PD31	**FCC2: TxD[3]** UTOPIA 8	**SCC1: RXD**	GND			VDD
PD30	**FCC2: TxEnb** UTOPIA master	**FCC2: TxEnb** UTOPIA slave		**SCC1: TXD**		

Table 45-20. Port D Dedicated Pin Assignment (PPARD = 1) (continued)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or I/O if Specified)	Default Input
PD29	**SCC1: $\overline{\text{RTS}}$**			**FCC1: RxAddr[3]** MPHY, master, multiplexed polling	**FCC1: RxAddr[3]** MPHY, slave, multiplexed polling **FCC1: RxClav2** MPHY, master, direct polling	GND
PD25	FCC2: TxClav UTOPIA, slave	**SCC3: RX/ USB: Rxd**	GND		**FCC2: TxClav** UTOPIA, master	
PD24	**SCC3: TXD/ USB: TN**			**FCC2: RxEnb** UTOPIA, master	**FCC2: RxEnb** UTOPIA, slave	
PD23	**SCC3: $\overline{\text{RTS}}$** **USB: TP**			**FCC2: RxAddr[1]** MPHY, master	**FCC2: RxAddr[1]** MPHY, slave	
PD22	**FCC2: TxD[2]** UTOPIA 8	**SCC4: RXD**	GND		**TDM_A2: L1TXD** I/O	GND
PD21	**SCC4: TXD**	**FCC2: RxD[3]** UTOPIA 8			**TDM_A2: L1RXD** I/O	GND
PD20	**SCC4: $\overline{\text{RTS}}$**	**FCC2: RxD[2]** UTOPIA 8			**TDM_A2: L1RSYNC**	GND
PD19	**FCC1: TxAddr[4]** MPHY, master, multiplexed polling	**FCC1: TxAddr[4]** MPHY, slave, multiplexed polling **FCC1: TxClav3** MPHY, master, direct polling	GND	**BRG1: BRGO**	**SPI: $\overline{\text{SPISEL}}$** (primary option)	PC25
PD18	**FCC1: RxAddr[4]** MPHY, master, multiplexed polling	**FCC1: RxAddr[4]** MPHY, slave, multiplexed polling **FCC1: RxClav3** MPHY, master, direct polling	GND		**SPI: SPICLK** I/O (primary option)	PC28
PD17	**BRG2: BRGO**	**FCC1: RxPrty** UTOPIA (primary option)	PC27		**SPI: SPIMOSI** I/O	VDD
PD16	**FCC1: TxPrty** UTOPIA				**SPI: SPIMISO** I/O	SPIMOSI
PD15	**TDM_C2: $\overline{\text{L1RQ}}$**	**FCC2: RxD[1]** UTOPIA 8			**I2C: I2CSDA** I/O	VDD

Table 45-20. Port D Dedicated Pin Assignment (PPARD = 1) (continued)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or I/O if Specified)	Default Input
PD14	**TDM_C2: L1CLKO**	**FCC2: RxD[0]** UTOPIA 8			**I2C: I2CSCL** I/O	GND
PD7		**SMC2: SMSYN**	GND	**FCC1: TxAddr[3]** MPHY, master, multiplexed polling	**FCC1: TxAddr[3]** MPHY, slave, multiplexed polling **FCC1: TxClav2** MPHY, master, direct polling	GND

45.6 Interrupts from Port C

The port C lines associated with $\overline{\text{CD}}x$ and $\overline{\text{CTS}}x$ have a mode of operation where the pin can be internally connected to the SCC/FCC but can also generate interrupts. Port C still detects changes on the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins and asserts the corresponding interrupt request, but the SCC/FCC simultaneously uses $\overline{\text{CTS}}$ and/or $\overline{\text{CD}}$ to automatically control operation. This lets the user fully implement protocols V.24, X.21, and X.21 bis (with the assistance of other general-purpose I/O lines).

To configure a port C pin as a $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ pin that connects to the SCC/FCC and generates interrupts, these steps should be followed:

1. Write the corresponding PPARC bit with a 1 and PSORC bit with 0.
2. Write the corresponding PDIRC bit with a zero.
3. Set the SIEXR bit (in the interrupt controller) to determine which edges cause interrupts.
4. Write the corresponding SIMR (mask register) bit with a 1 to allow interrupts to be generated to the core.
5. The pin value can be read at any time using PDATC.

NOTE

After connecting $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ to the SCC/FCC, the user must also choose the normal operation mode in GSMR[DIAG] to enable and disable SCC/FCC transmission and reception with these pins.

Appendix A
MPC8541E

This appendix describes the features of the MPC8541E, how it differs from the MPC8555E, and provides additional material and specifics for using this reference manual for the MPC8541E. It is intended to be used in conjunction with the *MPC8541E PowerQUICC™ III Integrated Processor Hardware Specifications* (MPC8541EEC).

A.1 MPC8541E Overview

Figure A-1 shows the major functional units within the MPC8541E.

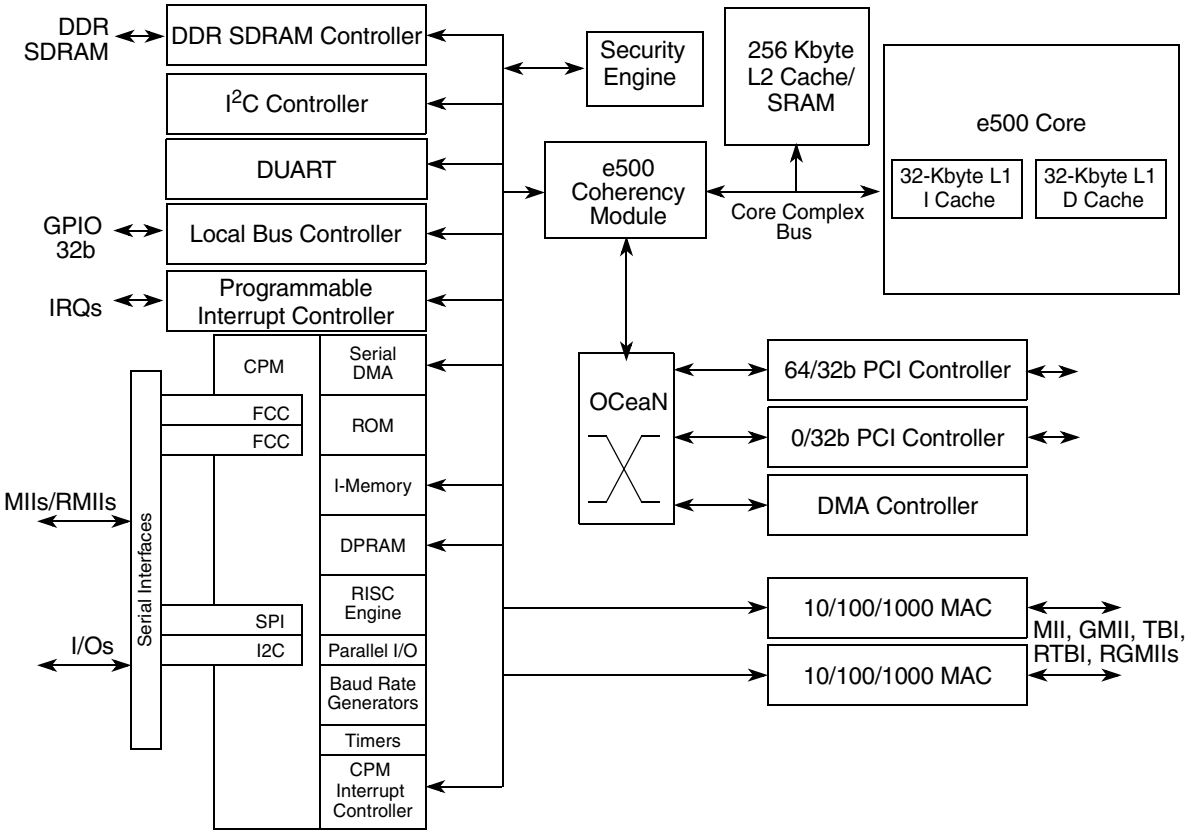


Figure A-1. MPC8541E Block Diagram

A.1.1 Key Features

The following lists an overview of the MPC8541E feature set.

- Embedded e500 core
  - High-performance, 32-bit core that implements the embedded category of the Power Architecture technology
  - Dual-issue superscalar, 7-stage pipeline design
  - 32-Kbyte L1 instruction cache and 32-Kbyte L1 data cache with parity protection
  - Lockable L1 caches—entire cache or on a per-line basis
  - Separate locking for instructions and data
  - Single-precision floating-point operations
  - Memory management unit especially designed for embedded applications
  - Enhanced hardware and software debug support
  - Dynamic power management
  - Performance monitor facility
- Integrated security engine (SEC)

  The SEC is optimized to handle all the algorithms associated with IPSec, SSL/TLS, SRTP, 802.11i standard, iSCSI, and IKE processing. The SEC 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
    - 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 execution unit (AESU)
    - Implements the Rijndael symmetric-key cipher
    - Key lengths of 128, 192, and 256 bits
    - 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- 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

- High-performance RISC CPM
  - — Two full-duplex fast communications controllers (FCCs) that support the following protocol:
    - IEEE802.3 standard/Fast Ethernet (10/100)
  - — Serial peripheral interface (SPI) support for master or slave
  - — I^2C bus controller
  - — General-purpose parallel ports—16 parallel I/O lines with interrupt capability
- 256 Kbytes of on-chip memory
  - — Can act as a 256-Kbyte level 2 cache
  - — Can act as a 256-Kbyte or two 128-Kbyte memory-mapped SRAM arrays
  - — Can be partitioned into 128-Kbyte L2 cache plus 128-Kbyte SRAM
  - — Full ECC support on a 64-bit boundary in both cache and SRAM modes
  - — SRAM operation supports relocation and is byte-accessible
  - — Cache mode supports instruction caching, data caching, or both
  - — External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types (stashing)
  - — Eight-way set-associative cache organization (1024 sets of 32-byte cache lines)
  - — Supports locking the entire cache or selected lines
    - Individual line locks set and cleared through instructions or by externally mastered transactions
  - — Global locking and Flash clearing done through writes to L2 configuration registers
  - — Instruction and data locks can be Flash cleared separately
  - — Read and write buffering for internal bus accesses
- Address translation and mapping unit (ATMU)
  - — Eight local access windows define mapping within local 32-bit address space
  - — Inbound and outbound ATMUs map to larger external address spaces
    - Three inbound windows plus a configuration window on PCI
    - Four outbound windows plus default translation for PCI
- DDR memory controller
  - — Programmable timing supporting first generation DDR SDRAM
  - — 64-bit data interface, up to 333-MHz data rate
  - — Four banks of memory supported, each up to 1 Gbyte

  - DRAM chip configurations from 64 Mbits to 1 Gbit with x8/x16 data ports
  - Full ECC support
  - Page mode support (up to 16 simultaneous open pages)
  - Contiguous or discontiguous memory mapping
  - Sleep mode support for self refresh DDR SDRAM
  - Supports auto refreshing
  - On-the-fly power management using CKE signal
  - Registered DIMM support
  - Fast memory access through JTAG port
  - 2.5-V SSTL2 compatible I/O
- Programmable interrupt controller (PIC)
  - Programming model is compliant with the OpenPIC architecture
  - Supports 16 programmable interrupt and processor task priority levels
  - Supports 12 discrete external interrupts
  - Supports 4 message interrupts with 32-bit messages
  - Supports connection of an external interrupt controller such as the 8259 programmable interrupt controller
  - Four global high resolution timers/counters that can generate interrupts
  - Supports additional internal interrupt sources
  - Supports fully nested interrupt delivery
  - Interrupts can be routed to external pin for external processing
  - Interrupts can be routed to the e500 core's standard or critical interrupt inputs
  - Interrupt summary registers allow fast identification of interrupt source
- Two I^2C controllers (one is contained within the CPM, the other is a stand-alone controller which is not part of the CPM)
  - Two-wire interface
  - Multiple master support
  - Master or slave I^2C mode support
  - On-chip digital filtering rejects spikes on the bus
- Boot sequencer
  - Optionally loads configuration data from serial ROM at reset through the stand-alone I^2C interface
  - Can be used to initialize configuration registers and/or memory
  - Supports extended I^2C addressing mode
  - Data integrity checked with preamble signature and CRC

- DUART
  - Two 4-wire interfaces (RXD, TXD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$)
  - Programming model compatible with the original 16450 UART and the PC16550D
- Local bus controller (LBC)
  - Multiplexed 32-bit address and data operating at up to 166 MHz
  - Eight chip selects support eight external slaves
  - Up to eight-beat burst transfers
  - The 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 bits)
- Two three-speed (10/100/1000) Ethernet controllers (TSECs)
  - Dual IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac standard compliant controllers
  - Support for Ethernet physical interfaces:
    - 10/100/1000 Mbps IEEE 802.3 standard GMII
    - 10/100 Mbps IEEE 802.3 standard MII
    - 10 Mbps IEEE 802.3 standard MII
    - 1000 Mbps IEEE 802.3z standard TBI
    - 10/100/1000 Mbps RGMII/RTBI
  - Full- and half-duplex support
  - Buffer descriptors are backward compatible with MPC8260 and MPC860T 10/100 programming models
  - 9.6-Kbyte jumbo frame support
  - RMON statistics support
  - 2-Kbyte internal transmit and receive FIFOs
  - MII management interface for control and status
  - Programmable CRC generation and checking
- OCeaN switch fabric
  - Three-port crossbar packet switch
  - Reorders packets from a source based on priorities
  - Reorders packets to bypass blocked packets

  - Implements starvation avoidance algorithms
  - Supports packets with payloads of up to 256 bytes
- Integrated DMA controller
  - Four-channel controller
  - All channels accessible by both local and remote masters
  - Extended DMA functions (advanced chaining and striding capability)
  - Support for scatter and gather transfers
  - Misaligned transfer capability
  - Interrupt on completed segment, link, list, and error
  - Supports transfers to or from any local memory or I/O port
  - Selectable hardware-enforced coherency (snoop/no-snoop)
  - Ability to start and flow control each DMA channel from external 3-pin interface
  - Ability to launch DMA from single write transaction
- PCI controllers
  - PCI 2.2 compatible
  - One 64-bit or two 32-bit PCI ports supported at 16 to 66 MHz
  - Host and agent mode support, 64-bit PCI port can be host or agent, if two 32-bit ports, only one can be an agent
  - 64-bit dual address cycle (DAC) support
  - Supports PCI-to-memory and memory-to-PCI streaming
  - Memory prefetching of PCI read accesses
  - Supports posting of processor-to-PCI and PCI-to-memory writes
  - PCI 3.3-V compatible
  - Selectable hardware-enforced coherency
  - Selectable clock source (SYSCLK or independent PCI_CLK)
- Power management
  - Fully static 1.2-V CMOS design with 3.3- and 2.5-V I/O
  - Supports power save modes: doze, nap, and sleep
  - Employs dynamic power management
  - Selectable clock source (SYSCLK or independent PCI_CLK)
- System performance monitor
  - Supports eight 32-bit counters that count the occurrence of selected events
  - Ability to count up to 512 counter-specific events
  - Supports 64 reference events that can be counted on any of the 8 counters
  - Supports duration and quantity threshold counting

  - Burstiness feature that permits counting of burst events with a programmable time between bursts
  - Triggering and chaining capability
  - Ability to generate an interrupt on overflow
- System access port
  - Uses JTAG interface and a TAP controller to access entire system memory map
  - Supports 32-bit accesses to configuration registers
  - Supports cache-line burst accesses to main memory
  - Supports large block (4-Kbyte) uploads and downloads
  - Supports continuous bit streaming of entire block for fast upload and download
- IEEE 1149.1 compliant, JTAG boundary scan
- 783-pin FC-PBGA package

The MPC8541E is pin compatible and functionally identical to the MPC8555E except for a reduced protocol set supported and lack of TDM (time division multiplexers) within the CPM. Figure A-2 shows the major functional units of the MPC8541E CPM.

The following list summarizes the major CPM features of the MPC8541E in relation to the MPC8555E:

- The communications processor (CP) is an embedded 32-bit RISC controller residing on a separate bus (CPM local bus). With this separate bus, the CP does not affect the performance of the e500 core. The CP handles the lower-layer tasks and DMA control activities, leaving the e500 core free to handle higher-layer activities. The CP has an instruction set optimized for communications, but that can also be used for general-purpose applications, relieving the system core of small, often repeated tasks. This is functionally identical with the MPC8555E.
- Two serial DMAs (SDMAs), one associated with the local bus and one associated with the e500 coherency module (ECM), handling transfers simultaneously. This is functionally identical with the MPC8555E.
- Two full-duplex, serial fast communications controllers (FCCs) supporting Fast Ethernet. This is functionally identical to the MPC8555E, however, the MPC8555E FCCs also support ATM and HDLC protocols.
- One full-duplex serial peripheral interface (SPI) providing a synchronous, character-oriented channel that supports a four-wire interface for communication with other microprocessors, peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices. This is functionally identical with the MPC8555E.
- I^2C bus controller providing communication with other I^2C capable devices. There are two I^2C controllers on the MPC8541E, this one in the CPM and a separate stand-alone I^2C controller as well. This is functionally identical with the MPC8555E.

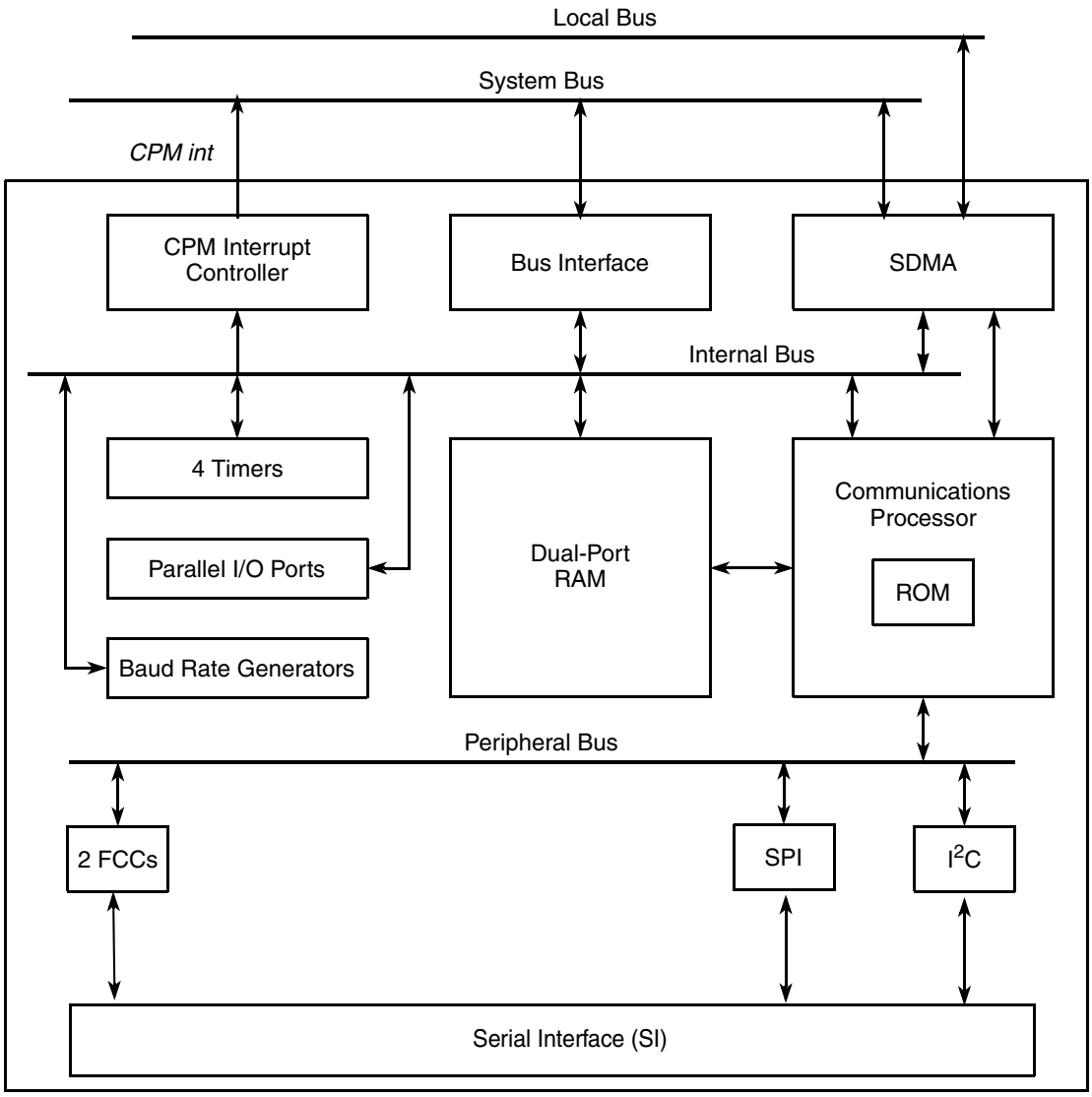


Figure A-2. MPC8541E CPM Block Diagram

A.2 How to Use This Book for the MPC8541E

By using the following MPC8541E CPM memory map (see Table A-1) and MPC8541E parallel I/O ports tables (beginning with Table A-2), and by ignoring those protocols in this reference manual that are not available in the MPC8541E, this reference manual is directly applicable to the MPC8541E.

A.2.1 MPC8541E CPM Memory Map

Table A-1 shows the CPM portion of the internal memory map.

Table A-1. MPC8541E Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
CPM Dual-Port RAM				
0x8_0000–0x8_1FFF	DPRAM1—Dual-port RAM	R/W	—	21.4/21-28
0x8_2000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_9FFF	DPRAM2—Dual-port RAM	R/W	—	21.4/21-28
0x8_A000–0x8_FFFF	Reserved	—	—	—
e500 Core Interface				
0x9_0000	CEAR—CPM error address register	R	0x0000_0000	21.2.3.1.1/21-18
0x9_0004	CEER—CPM error event register	R/W	0x0000	21.2.3.1.2/21-19
0x9_0006	CEMR—CPM error mask register	R/W	0x0000	21.2.3.1.3/21-20
SDMA				
0x9_0050	SMAER—System bus address error register	R	0x0000_0000	27.1.1/27-2
0x9_0054	Reserved	—	—	—
0x9_0058	SMEVR—System bus event register	R/W	0x0000_0000	27.1.2/27-2
0x9_005C	SMCTR—System bus control register	R/W	0x3800_0000	27.1.3/27-3
0x9_0060	LMAER—Local bus address error register	R	0x0000_0000	27.1.1/27-2
0x9_0064	Reserved	—	—	—
0x9_0068	LMEVR—Local bus event register	R/W	0x0000_0000	27.1.2/27-2
0x9_006C	LMCTR—Local bus control register	R/W	0x3800_0000	27.1.3/27-3
Interrupt Controller				
0x9_0C00	SICR—CPM interrupt configuration register	R/W	0x0000_0000	22.5.1.1/22-9
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIVEC—CPM interrupt vector register	R/W	0x0000_0000	22.5.1.5/22-14
0x9_0C08	SIPNR_H—CPM interrupt pending register (high)	R/W	0x0000_0000	22.5.1.3/22-11
0x9_0C0C	SIPNR_L—CPM interrupt pending register (low)	R/W	0x0000_0000	22.5.1.3/22-11
0x9_0C10	Reserved	—	—	—
0x9_0C14	SCPRR_H—CPM interrupt priority register (high)	R/W	0x0530_9770	22.5.1.2/22-10

Table A-1. MPC8541E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_0C18	SCPRR_L—CPM interrupt priority register (low)	R/W	0x0530_9770	22.5.1.2/22-10
0x9_0C1C	SIMR_H—CPM interrupt mask register (high)	R/W	0x0000_0000	22.5.1.4/22-12
0x9_0C20	SIMR_L—CPM interrupt mask register (low)	R/W	0x0000_0000	22.5.1.4/22-12
0x9_0C24	SIEXR—CPM external interrupt control register	R/W	0x0000_0000	22.5.1.6/22-15
0x9_0C28–0x9_0C7F	Reserved	—	—	—
Clock				
0x9_0C80	SCCR—System clock control register	R/W	0x0000_0000	25.1/25-2
Input/Output Port				
0x9_0D00	PDIRA—Port A data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D04	PPARA—Port A pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D08	PSORA—Port A special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D0C	PODRA—Port A open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D10	PDATA—Port A data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D14–0x9_0D1F	Reserved	—	—	—
0x9_0D20	PDIRB—Port B data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D24	PPARB—Port B pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D28	PSORB—Port B special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D2C	PODRB—Port B open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D30	PDATB—Port B data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	PDIRC—Port C data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D44	PPARC—Port C pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D48	PSORC—Port C special options register	R/W	0x0000_0000	45.2.5/45-13
0x9_0D4C	PODRC—Port C open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D50	PDATC—Port C data register	R/W	0x0000_0000	45.2.2/45-4
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	PDIRD—Port D data direction register	R/W	0x0000_0000	45.2.2.2/45-5
0x9_0D64	PPARD—Port D pin assignment register	R/W	0x0000_0000	45.2.4/45-10
0x9_0D68	PSORD—Port D special options register	R/W	0x0000_0000	45.2.5/45-13

Table A-1. MPC8541E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_0D6C	PODRD—Port D open drain register	R/W	0x0000_0000	45.2.1/45-1
0x9_0D70	PDATD—Port D data register	R/W	0x0000_0000	45.2.2/45-4
CPM Timers				
0x9_0D80	TGCR1—Timer 1 and timer 2 global configuration register	R/W	0x00	26.2.2/26-3
0x9_0D81	Reserved	—	—	—
0x9_0D84	TGCR2—Timer 3 and timer 4 global configuration register	R/W	0x00	26.2.2/26-3
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	TMR1—Timer 1 mode register	R/W	0x0000	26.2.3/26-5
0x9_0D92	TMR2—Timer 2 mode register	R/W	0x0000	26.2.3/26-5
0x9_0D94	TRR1—Timer 1 reference register	R/W	0x0000	26.2.4/26-7
0x9_0D96	TRR2—Timer 2 reference register	R/W	0x0000	26.2.4/26-7
0x9_0D98	TCR1—Timer 1 capture register	R/W	0x0000	26.2.5/26-7
0x9_0D9A	TCR2—Timer 2 capture register	R/W	0x0000	26.2.5/26-7
0x9_0D9C	TCN1—Timer 1 counter	R/W	0x0000	26.2.6/26-7
0x9_0D9E	TCN2—Timer 2 counter	R/W	0x0000	26.2.6/26-7
0x9_0DA0	TMR3—Timer 3 mode register	R/W	0x0000	26.2.3/26-5
0x9_0DA2	TMR4—Timer 4 mode register	R/W	0x0000	26.2.3/26-5
0x9_0DA4	TRR3—Timer 3 reference register	R/W	0x0000	26.2.4/26-7
0x9_0DA6	TRR4—Timer 4 reference register	R/W	0x0000	26.2.4/26-7
0x9_0DA8	TCR3—Timer 3 capture register	R/W	0x0000	26.2.5/26-7
0x9_0DAA	TCR4—Timer 4 capture register	R/W	0x0000	26.2.5/26-7
0x9_0DAC	TCN3—Timer 3 counter	R/W	0x0000	26.2.6/26-7
0x9_0DAE	TCN4—Timer 4 counter	R/W	0x0000	26.2.6/26-7
0x9_0DB0	TER1—Timer 1 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB2	TER2—Timer 2 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB4	TER3—Timer 3 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB6	TER4—Timer 4 event register	R/W	0x0000	26.2.7/26-8
0x9_0DB8–0x9_12FF	Reserved	—	—	—
FCC1				
0x9_1300	GFMR1—FCC1 general mode register	R/W	0x0000_0000	37.2/37-3

Table A-1. MPC8541E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1304	FPSMR1—FCC1 protocol-specific mode register	R/W	0x0000_0000	(Ethernet) 40.18.2/40-20
0x9_1308	FTODR1—FCC1 transmit on demand register	R/W	0x0000	37.6/37-8
0x9_130A	Reserved	—	—	—
0x9_130C	FDSR1—FCC1 data synchronization register	R/W	0x7E7E	37.5/37-7
0x9_130E	Reserved	—	—	—
0x9_1310	FCCE1—FCC1 event register	R/W	0x0000_0000	(Ethernet) 40.18.3/40-22
0x9_1312	Reserved	—	—	—
0x9_1314	FCCM1—FCC1 mask register	R/W	0x0000_0000	(Ethernet) 40.18.3/40-22
0x9_1316–0x9_131F	Reserved	—	—	—
FCC2				
0x9_1320	GFMR2—FCC2 general mode register	R/W	0x0000_0000	37.2/37-3
0x9_1324	FPSMR2—FCC2 protocol-specific mode register	R/W	0x0000_0000	(Ethernet) 40.18.2/40-20
0x9_1328	FTODR2—FCC2 transmit on-demand register	R/W	0x0000	37.6/37-8
0x9_132A	Reserved	—	—	—
0x9_132C	FDSR2—FCC2 data synchronization register	R/W	0x7E7E	37.5/37-7
0x9_132E	Reserved	—	—	—
0x9_1330	FCCE2—FCC2 event register	R/W	0x0000_0000	(Ethernet) 40.18.3/40-22
0x9_1332	Reserved	—	—	—
0x9_1334	FCCM2—FCC2 mask register	R/W	0x0000_0000	(Ethernet) 40.18.3/40-22
0x9_1336–0x9_138C	Reserved	—	0x00	—
FCC1 (continued)				
0x9_1390	GFEMR1—General FCC1 expansion mode register	R/W	0x00	37.3/37-7
0x9_1391–0x9_13AF	Reserved	—	—	—
FCC2 (continued)				
0x9_13B0	GFEMR2—General FCC2 expansion mode register	R/W	0x00	37.3/37-7

Table A-1. MPC8541E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_13B1–0x9_15EF	Reserved	—	—	—
BRGs 5–8				
0x9_15F0	BRGC5—BRG5 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15F4	BRGC6—BRG6 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15F8	BRGC7—BRG7 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_15FC	BRGC8—BRG8 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_1600–0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I2MOD—I^2C mode register	R/W	0x00	44.4.1/44-6
0x9_1861	Reserved	—	—	—
0x9_1864	I2ADD—I^2C address register	R/W	0x00	44.4.2/44-7
0x9_1865	Reserved	—	—	—
0x9_1868	II2BRG—I^2C BRG register	R/W	0x00	44.4.3/44-7
0x9_1869	Reserved	—	—	—
0x9_186C	I2COM—I^2C command register	R/W	0x00	44.4.5/44-8
0x9_186D	Reserved	—	—	—
0x9_1870	I2CER—I^2C event register	R/W	0x00	44.4.4/44-7
0x9_1871	Reserved	—	—	—
0x9_1874	I2CMR—I^2C mask register	R/W	0x00	44.4.4/44-7
0x9_1875–0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	CPCR—Communications processor command register	R/W	0x0000_0000	21.3.1/21-24
0x9_19C4	RCCR—CP configuration register	R/W	0x0000_0000	21.2.6/21-22
0x9_19C8–0x9_19D5	Reserved	—	—	—
0x9_19D6	RTER—CP timers event register	R/W	0x0000	21.5.4/21-35
0x9_19DA	RTMR—CP timers mask register	R/W	0x0000	21.5.4/21-35
0x9_19DC	RTSCR—CP time-stamp timer control register	R/W	0x0000	21.2.7/21-23
0x9_19DE	Reserved	—	—	—
0x9_19E0	RTSR—CP time-stamp register	R/W	0x0000_0000	21.2.8/21-24

Table A-1. MPC8541E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
BRGs 1–4				
0x9_19F0	BRGC1—BRG1 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19F4	BRGC2—BRG2 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19F8	BRGC3—BRG3 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_19FC	BRGC4—BRG4 configuration register	R/W	0x0000_0000	25.2/25-3
0x9_1A00–0x9_1A9F	Reserved	—	—	—
SPI				
0x9_1AA0	SPMODE—SPI mode register	R/W	0x0000	43.4.1/43-6
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPIE—SPI event register	R/W	0x00	43.4.2/43-9
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPIM—SPI mask register	R/W	0x00	43.4.2/43-9
0x9_1AAB	Reserved	—	—	—
0x9_1AAD	SPCOM—SPI command register	W	0x00	43.4.3/43-10
0x9_1AA7–0x9_1B00	Reserved	—	—	—
CPM Mux				
0x9_1B02	CMXSI2CR—CPM mux SI2 clock route register	R/W	0x00	24.4.2/24-8
0x9_1B03	Reserved	—	—	—
0x9_1B04	CMXFCR—CPM mux FCC clock route register	R/W	0x0000_0000	24.4.3/24-8
0x9_1B08–0x9_1B46	Reserved	—	—	—
SI2 Registers				
0x9_1B48	SI2GMR—SI2 global mode register	R/W	0x00	23.6.1/23-14
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2CMDR—SI2 command register	R/W	0x00	23.6.4/23-20
0x9_1B4B	Reserved	—	—	—
0x9_1B4C	SI2STR—SI2 status register	R/W	0x00	23.6.5/23-21
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2RSR—SI2 RAM shadow address register	R/W	0x0000	23.6.3/23-20
0x9_1B50–0x9_1FFF	Reserved	—	—	—

Table A-1. MPC8541E Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
SI2 RAM				
0x9_2800–0x9_29FF	SI2TxRAM—SI 2 transmit routing RAM	—	—	23.5.3/23-9
0x9_2A00–0x9_2BFF	Reserved	—	—	—
0x9_2C00–0x9_2DFF	SI2RxRAM—SI 2 receive routing RAM	—	—	23.5.3/23-9
0x9_2E00–0x9_3FFF	Reserved	—	—	—
Instruction RAM				
0xA_0000–0xA_0FFF	Dual-port RAM (instruction RAM only)	—	—	21.4/21-28

A.2.2 MPC8541E Parallel I/O Port Pin Assignments

The functional design of the MPC8541E parallel I/O ports is identical to the MPC8555E. However, as stated previously, the protocols supported by each device differ, and the MPC8541E does not support TDMs. The following MPC8541E parallel I/O pin assignments are direct replacement tables for those given for the MPC8555E, in Section 45.5, "Port Tables."

Table A-2 shows the MPC8541E port A pin assignments.

Table A-2. Port A Dedicated Pin Assignment (PPARA = 1)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or I/O if Specified)	Default Input
PA31			GND		**FCC1: COL** MII	GND
PA30			GND		**FCC1: CRS** MII	GND
PA29				**FCC1: TX_ER** MII		
PA28			GND	**FCC1: TX_EN** MII/RMII		
PA27			GND		**FCC1: RX_DV** MII **FCC1: CRS_DV** RMII	GND
PA26			GND		**FCC1: RX_ER** MII/RMII	GND

Table A-2. Port A Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or I/O if Specified)	Default Input
PA25				MSNUM[0][1]		
PA24				MSNUM[1] [1]		
PA23						
PA22						
PA21	**FCC1: TxD[3]** MII nibble					
PA20	**FCC1: TxD[2]** MII nibble					
PA19	**FCC1: TxD[1]** MII nibble **FCC1: TxD[1]** **RMII dibit**					
PA18	**FCC1: TxD[0]** MII nibble **FCC1: TxD[0]** RMII dibit					
PA17		**FCC1: RxD[0]** MII nibble **FCC1: RxD[0]** **RMII dibit**	GND			
PA16		**FCC1: RxD[1]** MII nibble **FCC1: RxD[1]** **RMII dibit**	GND			
PA15		**FCC1: RxD[2]** MII nibble	GND			
PA14		**FCC1: RxD[3]** MII nibble	GND			
PA13			GND	MSNUM[2] [1]		
PA12			GND	MSNUM[3] [1]		
PA11			GND	MSNUM[4] [1]		
PA10			GND	MSNUM[5] [1]		
PA9						
PA8			GND			

[1] MSNUM[0–5] is the sub-block code of the peripheral controller using SDMA; MSNUM[5] indicates which section, transmit or receive, is active during the transfer. See Section 27.1.2, “SDMA Event Registers (SMEVR, LMEVR).”

Table A-3 shows the MPC8541E port B pin assignments.

Table A-3. Port B Dedicated Pin Assignment (PPARB = 1)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or I/O if Specified)	Default Input
PB31	**FCC2: TX_ER** MII					GND
PB30		**FCC2: RX_DV** MII **FCC2: CRS_DV** RMII	GND			GND
PB29				**FCC2: TX_EN** MII/RMII		GND
PB28		**FCC2: RX_ER** MII/RMII	GND			GND
PB27		**FCC2: COL** MII	GND			GND
PB26		**FCC2: CRS** MII	GND			GND
PB25	**FCC2: TxD[3]** MII nibble					GND
PB24	**FCC2: TxD[2]** MII nibble					GND
PB23	**FCC2: TxD[1]** MII nibble **FCC2: TxD[1]** RMII dibit					GND
PB22	**FCC2: TxD[0]** MII nibble **FCC2: TxD[0]** RMII dibit					GND
PB21		**FCC2: RxD[0]** MII nibble **FCC2: RxD[0]** RMII dibit	GND			GND
PB20		**FCC2: RxD[1]** MII nibble **FCC2: RxD[1]** RMII dibit	GND			GND
PB19		**FCC2: RxD[2]** MII nibble	GND			
PB18		**FCC2: RxD[3]** MII nibble	GND			

Table A-4 shows the MPC8541E port C pin assignments.

Table A-4. Port C Dedicated Pin Assignment (PPARC = 1)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or I/O if Specified)	Default Input
PC29	**BRG2: BRGO**	CLK3/TIN2	CLK7			GND
PC28	**Timer2: $\overline{\text{TOUT}}$**	CLK4/TIN1	CLK8		**SPI: SPICLK1** I/O (secondary option)	GND
PC27	**Timer1: $\overline{\text{TOUT}}$**	CLK5	GND	**BRG3: BRGO**		GND
PC26	**Timer3: $\overline{\text{TOUT}}$**	CLK6	GND			BRGO1
PC25		CLK7/TIN4	GND	**BRG4: BRGO**	**SPI: $\overline{\text{SPISEL}}$1** (secondary option)	VDD
PC24	**BRG1: BRGO**	CLK8/TIN3	GND	**Timer4: $\overline{\text{TOUT}}$**		GND
PC23	**BRG5: BRGO**	CLK9	CLK13			GND
PC22		CLK10	CLK14			
PC21	**BRG6: BRGO**	CLK11	CLK15			GND
PC20		CLK12	CLK16			
PC19	**BRG7: BRGO**	CLK13	GND		**timer1/2: $\overline{\text{TGATE1}}$**	GND
PC18		CLK14	GND		**timer3/4: $\overline{\text{TGATE2}}$**	GND
PC17	**BRG8: BRGO**	CLK15	GND			VDD
PC16		CLK16	GND			
PC15			by PC29			GND
PC14			by PC23			GND
PC13	**BRG5: BRGO**					GND
PC12			VDD			GND
PC11			by PC8			
PC10			GND			
PC9			GND			GND
PC8			GND	**SI2: L1ST1** Strobe		GND
PC7			GND			GND
PC6	**SI2: L1ST2** Strobe		GND			GND

Table A-4. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or I/O if Specified)	Default Input
PC5				**SI2: L1ST3** Strobe		GND
PC4			GND	**SI2: L1ST4** Strobe		GND
PC1	**BRG6: BRGO**		GND			GND
PC0	**BRG7: BRGO**		GND			GND

Table A-5 shows the MPC8541E port D pin assignments.

Table A-5. Port D Dedicated Pin Assignment (PPARD = 1)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or I/O if Specified)	Default Input
PD31			GND			VDD
PD30						
PD29						GND
PD25			GND			
PD24						
PD23						
PD22			GND			GND
PD21						GND
PD20						GND
PD19			GND	**BRG1: BRGO**	**SPI: $\overline{\text{SPISEL}}$** (primary option)	PC25
PD18			GND		**SPI: SPICLK** I/O (primary option)	PC28
PD17	**BRG2: BRGO**		PC27		**SPI: SPIMOSI** I/O	VDD
PD16					**SPI: SPIMISO** I/O	SPIMOSI

Table A-5. Port D Dedicated Pin Assignment (PPARD = 1) (continued)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or I/O if Specified)	Default Input
PD15					**I2C: I2CSDA** I/O	VDD
PD14					**I2C: I2CSCL** I/O	GND
PD7			GND			GND

A.2.3 Chapter Advisory for the MPC8541E

The following chapters in this manual do not apply to the MPC8541E and should be ignored:

- Chapters 28–33, concerning SCCs
- Chapter 34, "QUICC Multi-Channel Controller (QMC)"
- Chapter 35, "Universal Serial Bus Controller"
- Chapter 36, "Serial Management Controllers (SMCs)"
- Chapters 38 and 39, concerning FCC protocols not supported on the MPC8541E
- Chapters 41 and 42, concerning ATM, which is not supported on the MPC8541E

The following chapters refer to implementation that differs from the MPC8541E:

- Chapter 23, "Serial Interface with Time-Slot Assigner"—There are no time-slot assigners on the MPC8541E, so references to them should be ignored. Also, all material concerning serial protocols not implemented on the MPC8541E should be ignored.
- Chapter 24, "CPM Multiplexing"—All material concerning serial protocols not implemented on the MPC8541E should be ignored.
- Chapter 37, "Fast Communications Controllers (FCCs)"—The only protocol supported on the MPC8541E FCCs is Fast Ethernet. References to all other protocols in this chapter should be ignored. Users may find that Chapter 40, "CPM Fast Ethernet Controller," contains everything needed to implement Fast Ethernet on the FCCs.

Appendix B
Revision History

This appendix provides a list of the major differences between the *MPC8555E PowerQUICC™ III Integrated Processor Reference Manual*, Revision 0 through Revision 2.

B.1 Changes from Revision 1 to Revision 2

Major changes to the *MPC8555E PowerQUICC™ III Integrated Processor Reference Manual*, from Revision 1 to Revision 2 are as follows:

Section, Page	Changes
Book	Updated references to PowerPC, Book E, AIM, EIS, etc. technology.
Chapter 1, 1-1	Replace all the paragraphs on page 1-1 with the following:

Freescale Semiconductor's MPC8555E PowerQUICC™ III integrated communications processor includes a wide range of advanced Freescale technologies, modular cores, and peripherals. Leveraging Freescale's system-on-chip (SoC) PowerQUICC III platform architecture, the MPC8555E combines the powerful Book E e500 core and communications peripheral technology to balance processor performance with I/O system throughput. The processor is designed to offer clock speeds scaling from 533 MHz to 1 GHz.

This chapter provides a high-level description of features and functionality of the MPC8555E and MPC8541E integrated communications processors. It is written from the perspective of the MPC8555E, which is the superset device. For specifics on how to use this manual for the MPC8541E, see Appendix A, “MPC8541E.”

1.1 Introduction

Freescale's MPC8555E device integrates two processing blocks: a high-performance e500 core that implements the Power Architecture™ definition of the Book E instruction-set architecture and a RISC-based communications processor module (CPM) that supports a wide range of communications peripherals. This innovative architecture is designed to reduce power consumption and offer a more balanced approach to processing than traditional processor architectures. The CPM offloads low-level peripheral communications tasks, enabling the embedded e500 core to manage high-level processing tasks. The MPC8555E device's high level of integration helps simplify board design and enhances system-level bandwidth and performance. In addition to the e500 core and CPM, the MPC8555E features an integrated security engine, a double data rate SDRAM (DDR SDRAM) memory controller, dual Gigabit Ethernet controllers, a four-channel DMA controller, dual asynchronous receiver/transmitters (DUART), and a 64-bit PCI controller that can also serve as two 32-bit PCI ports. Dual on-chip PCI support provides a cost-effective alternative to separate, discrete PCI bridges and chipsets for I/O-intensive

applications that require multiple PCI interfaces. In addition to these features, the MPC8555E provides a local bus controller and I^2C support.

The MPC8555E processor features a security engine that supports DES, 3DES, MD-5, SHA-1, AES, and ARC-4 encryption algorithms, as well as offering a public key accelerator and on-chip random number generator. This embedded security core is derived from Freescale's security coprocessor product line and offers the same direct-memory access (DMA) and parallel processing capabilities, as well as the ability to perform single-pass encryption and authentication as required by widely used security protocols, such as IPsec and 802.11i. Integrated security makes the MPC8555E an optimal communications processor solution for applications that require security features in concert with high performance and low system-level cost.

1.3.1, 1-10 Replace the second bullet with the following:

- Implements additional instructions, registers, and interrupts defined by APUs. The SPE provides an extensive instruction set for 64-bit vector integer, single-precision floating-point, and fractional operations. The SPFP APU provides scalar (32-bit) single-precision floating-point instructions.

NOTE

The SPE APU and SPFP APU functionality will be implemented in all PowerQUICC III devices. However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or SPFP APU instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale offers a libcfsl_e500 library that uses SPE APU instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

1.3.3, 1-16 In the first bullet, replace the first sentence with the following:

Two full-duplex, serial fast communications controllers (FCCs) supporting ATM (155 Mbps) protocol through two UTOPIA level II interfaces.

2.2.3.2, 2-6 In Figure 2-2, the bit field ends at 15, this should be changed to 31.

2.3.1, 2-10 Insert the following new Section 2.3.1 and, subsequently, renumber existing sections.

2.3.1 Accessing CCSR Memory from the e500 Core

When the local e500 processor is used to configure CCSR space, the CCSR memory space should typically be marked as cache-inhibited and guarded.

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

To guarantee that the results of any sequence of writes to configuration registers are in effect, the final configuration register write should be 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.

2.4, 2-56 In Table 2-9, replace the Offset 0xE_0000 row with the following:

0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn_n*1*nn*	18.4.1.1/18-4

3.1, 3-1 In Figure 3-1, LSDAMUX is incorrectly shown muxed with LGPL5. There is no LSDAMUX signal; LGPL5 should appear by itself.

3.2, 3-16 In Table 3-3, LSDAMUX is incorrectly shown muxed with LGPL5. There is no LSDAMUX signal; LGPL5 should appear by itself.

4.4.2, 4-10 Correct the first sentence of the note associated with sequence step four to read as follows:

If the JTAG signals are not used, $\overline{\text{TRST}}$ may be tied active; however, it is recommended that $\overline{\text{TRST}}$ not remain asserted after negation of $\overline{\text{HRESET}}$.

Modify the second paragraph of the note as follows:

There is no need to assert the $\overline{\text{SRESET}}$ signal when $\overline{\text{HRESET}}$ is asserted. If $\overline{\text{SRESET}}$ is asserted on negation of $\overline{\text{HRESET}}$, the POR sequence will be paused after the e500 core PLL is locked and before the e500 reset is negated. The POR sequence will be resumed when $\overline{\text{SRESET}}$ is negated.

4.4.3, 4-11 Replace the second paragraph with the following:

All POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board.

4.4.3.10, 4-18 Replace the third sentence with the following:

Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 18.4.1.1, “POR PLL Status Register (PORPLLSR).”

5.1.1, 5-3 Replace the second paragraph of the note with the following:

Freescale Semiconductor offers a libcfsl_e500 library that uses SPE instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

5.2, 5-5 Replace Table 5-1 with the following:

MPC8555E Revision	Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
1.1	2.0	0x8020_0020	0x8079-0011 (MPC8555E) 0x8071-0011 (MPC8555) 0x807A-0011 (MPC8541E) 0x8072-0011 (MPC8541)

5.6, 5-18 In Figure 5-6, remove DVC1 and DVC2 registers.

5-14, 5-32 — In Table 5-8, replace the last two sentences in the description of SPE and SPFP APUs with the following:

Freescale offers a libcfsl_e500 library that uses SPE APU instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

In Table 5-8, the description of HID1 fields should appear as follows:

HID1 Implementation	PLL_MODE. Set to 01 PLL_CFG. PowerQUICC III devices support the following: 0001_00 Ratio of 2:1 0001_01 Ratio of 5:2 (2.5:1) 0001_10 Ratio of 3:1 0001_11 Ratio of 7:2 (3.5:1) NEXEN, R1DPE, R2DPE, MPXTT, MSHARS, SSHAR, ATS, and MID are not implemented On PowerQUICC III devices, ABE must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.
	Please refer to the description of HID1[RFXE] in Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” If RFXE is 0, conditions that cause the assertion of *core_fault_in* cannot directly cause the e500 to generate a machine check; however, PowerQUICC III devices must be configured to detect and enable such conditions. The following describes how error bits should be configured: • ECM mapping errors: EEER[LAEE] must be set. See Section 8.2.1.4, “ECM Error Enable Register (EEER).” • L2 multiple-bit ECC errors: L2ERRDIS[MBECCDIS] must be cleared to ensure that error can be detected. L2ERRINTEN[MBECCINTEN] must be set. See Section 7.3.1.5, “L2 Error Registers.” • DDR multiple-bit ECC errors. ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and DDR_SDRAM_CFG[ECC_EN] must be one to ensure that an interrupt is generated. See Section 9.4.1, “Register Descriptions.” • PCI. The appropriate parity detect and master-abort bits in ERR_DR must be cleared and the corresponding enable bits in ERR_EN must be set to ensure that an interrupt is generated. • Local bus controller parity errors. LTEDR[PARD] must be cleared and LTEIR[PARI] must be set to ensure that an parity errors can generate an interrupt. See Section 13.3.1.11, “Transfer Error Check Disable Register (LTEDR),” and Section 13.3.1.12, “Transfer Error Interrupt Enable Register (LTEIR).”

6.1.1, 6-2 — In Figure 6-1, remove DVC1 and DVC2 registers.

6.5.4, 6-14 — In Figure 6-9, correct the system version register reset value to read as follows:

0x8079_0011 (for MPC8555E) / 0x807A_0011 (for MPC8541E)
0x8071_0011 (for MPC8555) / 0x8072_0011 (for MPC8541)

6.7.2.4, 6-22 — Remove MCSR[47] (GL_CI) from Figure 6-26 and Table 6-12. This bit is not supported.

6.10.2, 6-26 Please replace the description of HID1[RFXE] in Table 6-19 with the following:

Table 6-19. HID1 Field Descriptions

Bits	Name	Description
46	RFXE	Read fault exception enable. Enables the core to internally generate a machine check interrupt when *core_fault_in* is asserted. Depending on the value of MSR[ME], this results in either a machine check interrupt or a checkstop. 0 Assertion of *core_fault_in* cannot cause a machine check. The core does not execute any instructions from a faulty instruction fetch and does not execute any load instructions that get their data from a faulty data fetch. On the e500v2, if these instructions are eventually required by the sequential programming model (that is, they are not in a speculative execution path), the e500v2 stalls until an asynchronous interrupt is taken. The e500v1 does not stall when faulty instructions or data are received, as described in the following note. **Note:** The e500v1 does not stall when faulty instructions or data are received. Instead, it continues processing with faulty instructions or data. The only reliable way to prevent such behavior is to set RFXE, which causes a machine check before the faulty instructions or data are used. To avoid the use of faulty instructions or data and to have good error determination, software must set RFXE and program the PIC to interrupt the processor when errors occur. As a result, software must deal with multiple interrupts for the same fundamental problem. 1 Assertion of *core_fault_in* causes a machine check if MSR[ME] = 1 or a checkstop if MSR[ME] = 0. The *core_fault_in* signal is asserted to the core when logic outside of the core has a problem delivering good data to the core. For example, the front-side L2 cache asserts *core_fault_in* when an ECC error occurs and ECC is enabled. As a second example, it is asserted when there is a master abort on a PCI transaction. See "Proper Reporting of Bus Faults," in the core complex bus chapter of the *PowerPC e500 Core Family Reference Manual*. The RFXE bit provides flexibility in error recovery. Typically, devices outside of the core have some way other than the assertion of *core_fault_in* to signal the core that an error occurred. Usually, this is done by channeling interrupt requests through a programmable interrupt controller (PIC) to the core. In these cases, the assertion of *core_fault_in* is used only to prevent the core from using bad data before receiving an interrupt from the PIC (for example, an external or critical input interrupt). Possible combinations of RFXE and PIC configuration are as follows: • RFXE = 0 and the PIC is configured to interrupt the processor. In this configuration, the assertion of *core_fault_in* does not trigger a machine check interrupt. The core does not use the faulty instructions or data and may stall. The PIC interrupts the core so that error recovery can begin. This configuration allows the core to query the PIC and the rest of the system for more information about the cause of the interrupt, and generally provides the best error recovery capabilities. • RFXE = 1 and the PIC is not configured to interrupt the processor. This configuration provides quick error detection without the overhead of configuring the PIC. When the PIC is not configured, setting RFXE avoids stalling the core when *core_fault_in* is asserted. Determination of the root cause of the problem may be somewhat more difficult than it would be if the PIC were enabled. • RFXE = 1 and the PIC is configured to interrupt the processor. In this configuration, the core may receive two interrupts for the same fundamental error. The two interrupts may occur in any order, which may complicate error handling. Therefore, this is usually not an interesting configuration for a single-core device. This may, however, be an interesting configuration for multi-core devices in which the PIC may steer interrupts to a processor other than the one that attempted to fetch the faulty data. • RFXE = 0 and the PIC is not configured to interrupt the processor. This is not a recommended configuration. The processor may stall indefinitely due to an unreported error.

6.13.1.1, 6-42 In Table 6-32, change the following DBCR0[RST] 1x description to read as follows:

1x Causes a hard reset if MSR[DE] and DBCR0[IDM] are set. Always cleared on subsequent cycle. This causes a hard reset to the core only.

7.3.1.5.2, 7-16 In Figure 7-17, "L2 Error Detect Register (L2ERRDET)," bit 0, MULL2ERR, is incorrectly shown as read only. The field should be w1c, like the other fields in this register.

9.4.1.7, 9-16 In Table 9-12, the description for REFINT, change the reference in the last sentence to Section 18.5.1.5.3.

9.4.1.8, 9-16 — Add the following note after the first paragraph:

NOTE

The DDR_SDRAM_CLK_CNTL[SS_EN] default value out of reset is 0 and this must be set to a 1 for proper operation.

10.1.3, 10-4 — In Table 10-1, change the description for the second bulleted item of the Reset core interrupt type to read as follows:

- core_hreset_req. Internal signal to the device, output from core, input to the platform—caused by writing to the core DBCR0[RST]. This condition is additionally qualified with MSR[DE] and DBCR0[IDM] bits. Note that assertion of this signal causes a hard reset of the core only.

10.2.2, 10-8 — In Table 10-5, remove the sentence, 'The timing requirements for edge-sensitive interrupts can be found in the *MPC8555E PowerQUICC™ III Integrated Processor Hardware Specifications*.' That timing is not provided in the hardware specifications.

11.3.1.2, 11-6 — Correct the following I2CFDR[FDR] values (SCL clock frequency divider ratios) in Table 11-5 (all other FDR value definitions remain unchanged):

FDR:	Divider (decimal):	FDR:	Divider (decimal):
0x00	384	0x20	256
0x01	416	0x21	288
0x02	480	0x22	320
0x03	576	0x23	352
0x04	640	0x24	384
0x05	704	0x25	448
0x06	832	0x26	512
0x07	1024	0x27	576

11.4.1.2, 11-12 — Replace the last sentence of the first paragraph with the following:

An I^2C device cannot be master and slave at the same time; if this is attempted, the results are boundedly undefined.

11.4.5, 11-17 — Add the following after the first sentence in the first paragraph:

The boot sequencer accesses the I^2C serial ROM at an interface frequency designated by the default value of the I2CFDR[FDR] field, 0x2C, which corresponds to a divider of 1280. See Section 11.3.2.1, "I^2C Frequency Divider Register (I2CFDR)," for additional details of the I2CFDR.

12.3.1.3, 12-7 Table 12-8, and the text that introduces it, should be replaced with the following:

Table 12-8 shows baud rate when the input clock is at certain frequencies. Note that because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

Table 12-8. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		Input Clock (CCB) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	1736	6C8	266	9600.61444	0.0064
19,200	868	364	266	19,201.22888	0.0064
38,400	434	1B2	266	38,402.45776	0.0064
56,000	298	12A	266	55,928.41163	0.1280
128,000	130	82	266	128,205.12821	0.1600
256,000	65	41	266	256,410.25641	0.1600
9,600	2170	87A	333	9600.61444	0.0064
19,200	1085	43D	333	19,201.22888	0.0064
38,400	543	21F	333	38,367.09638	0.0858
56,000	372	174	333	56,003.58423	0.0064
128,000	163	A3	333	127,811.86094	0.1472
256,000	81	51	333	257,201.64609	0.4672
9,600	3472	D90	533	9600.61444	0.0064
19,200	1736	6C8	533	19,201.22888	0.0064
38,400	868	364	533	38,402.45776	0.0064
56,000	595	253	533	56,022.40896	0.0400
128,000	260	104	533	128,205.12821	0.1600
256,000	130	82	533	256,410.25641	0.1600

Also in this section, the text between Table 12-8 and the end of the section should be removed.

13.1.2, 13-2 Correct the first sub-bullet under primary bullet, ‘Memory controller with eight memory banks’ to read as follows:

— 32-bit address decoding with mask

13.3.1.2.1, 13-12 Correct the first two sentences of the first paragraph to read as follows:

The address mask field of the option registers (OR*n*[AM]) mask up to 17 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.

13.3.1.2.1, 13-12 Remove the left-most column of Table 13-5.

13.3.1.10, 13-25 In the second paragraph, add the following sentence to the end of the paragraph:

Note that LTEATR[V] bit has to be cleared to register subsequent errors in LTESR.

13.3.1.13, 13-28	Replace the last sentence of the first paragraph with the following: After LTEATR[V] has been set, software must clear this bit to allow LBC error registers to update following any subsequent errors.
13.3.1.13, 13-28	In Figure 13-16 and Table 13-19, remove register field LTEATR[XA] and change bits 28–29 to be ‘Reserved’.
13.3.1.14, 13-29	In Table 13-20, correct the LTEAR[A] field description to read as follows: Transaction address for the error. Holds the 32-bit address of the transaction resulting in an error.
13.3.1.16, 13-31	In Table 13-22, add the following note to the description of LCRR[CLKDIV]: It is critical that no transactions are being executed via the local bus while CLKDIV is being modified. As such, prior to modification, the user must ensure that code is not executing out of the local bus. Once LCRR[CLKDIV] is written, the register should be read, and then an isync should be executed.
13.4, 13-33	In Figure 13-20, correct the label for the arrow in the upper left corner to read: ‘32-bit System Address’.
13.4.1.1, 13-33	Correct the third sentence of the first paragraph to read as follows: 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]).
13.4.1.7, 13-37	Replace the last sentence of the first paragraph with the following: Setting LTEDR[BMD] disables bus monitor error checking (that is, 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 13.4.4.1.4, “Exception Requests”) or terminate a GPCM access.
13.4.2.2.3, 13-44	The title of Figure 13-26 lists the OR*n* timing parameters incorrectly; CSNT applies only to write transactions, EHTR only to reads. The figure is also truncated and shows incorrect timing for LBCTL. Replace the figure and title with the following:

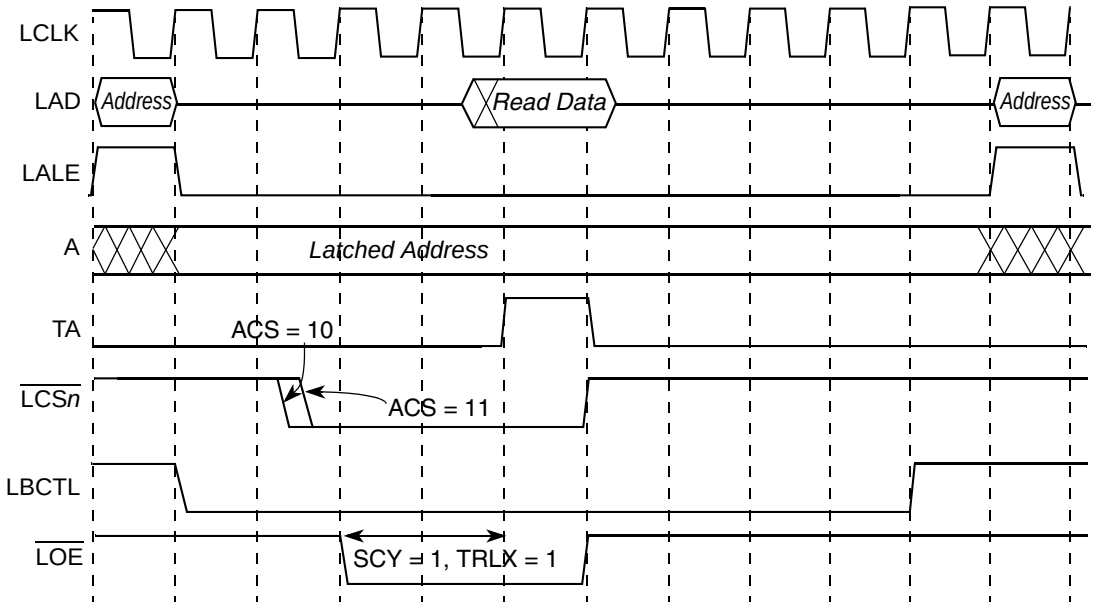


Figure 13-26. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, EHTR = 0, TRLX = 1)

13.4.2.4, 13-50 In Table 13-27, remove fields BR0[XBA] and OR0[XAM].

13.4.3.1, 13-50 Replace the last sentence of the first paragraph with the following:

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].

13.4.4.2, 13-65 Remove the third paragraph, beginning with 'Note that the UPM memory...'. Add the following note to the end of this section:

NOTE

In order to enforce proper ordering between updates to the MxMR register and the dummy accesses to the UPM memory region, two rules must be followed:

1. Since the result of any update to the MxMR register must be in effect before the dummy read or write to the UPM region, a write to MxMR should be followed immediately by a read of MxMR.
2. The UPM memory region should have the same MMU settings as the memory region containing the MxMR configuration register; both should be mapped by the MMU as Cache-Inhibited and Guarded. This prevents the e500 core from re-ordering a read of the UPM memory around the read of MxMR. Once 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.

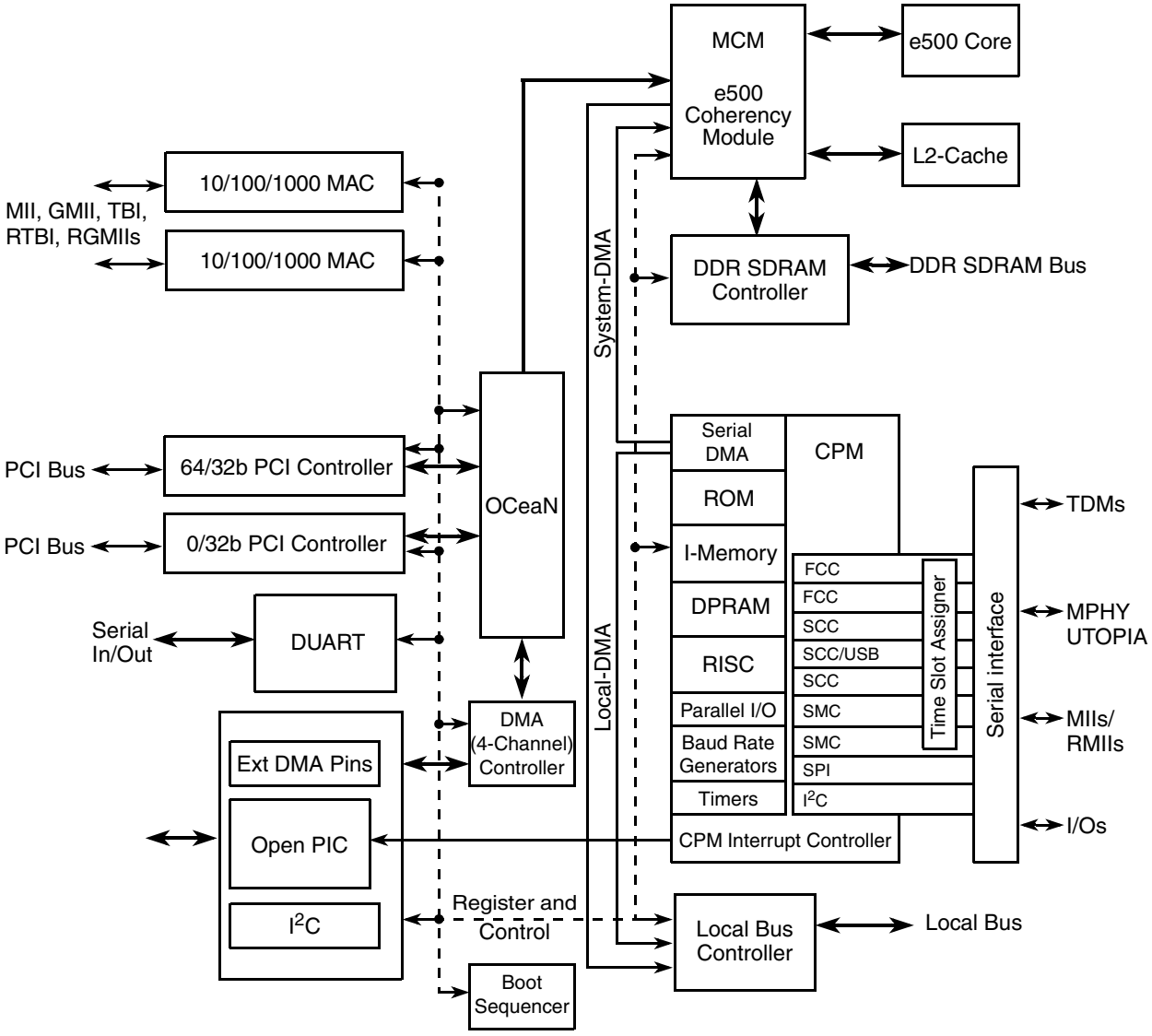
13.5.4.3.3, 13-94 In Table 13-39, remove fields BR*n*[XBA] and OR*n*[XAM].

13.5.4.3.5, 13-95	In Table 13-43, remove fields BR*n*[XBA] and OR*n*[XAM].
14.2, 14-7	Change the fifth element of the second bullet item in the TSEC features list to read: — 10/100 Mbps RGMII
14.5.3.2.1, 14-32	In Table 14-12, modify the FIFO_PAUSE_CTRL[TFC_PAUSE_EN] field description to read as follows: TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset but should always be set during initialization as undefined behavior results during normal operation when left cleared. 0 Pause control frame transmission disabled (default, but must be set during initialization). 1 Pause control frame transmission enabled.
14.5.3.9.1, 14-91	In Table 14-96, correct ATTR[ELCWT] and ATTR[BDLWT] field values of 11 to be ‘Reserved’.
14.5.4.2, 14-93	In Figure 14-100 and Table 14-99, correct default reset values of control register fields AN Enable, Full Duplex, and Speed_1 to all be zero.
14.5.4.6, 14-99	In Figure 14-104, correct the reset value of the TBI ANEX register to be 0000_0000_0000_0100.
14.6.1.1, 14-105	In Figure 14-110, superscript the ‘1’ on the EC_MDC and ECC_MDIO lines.
14.6.1.2, 14-106	In Figure 14-111, remove TSEC*n*_COL signal, since it is not used for a GMII interface.
14.6.1.2, 14-106	In Figure 14-111, make the following changes: The third line from the top, ‘Transmit Nibble Data,’ should be changed to ‘Transmit Data.’ The sixth line from the top, ‘Transmit Clock,’ should be removed. The seventh line from the top, ‘Receive Nibble Data,’ should be changed to ‘Receive Data.’ Superscript the ‘1’ on the EC_MDC and ECC_MDIO lines.
14.6.1.3, 14-107	In Figure 14-112, superscript the ‘1’ on the EC_MDC and ECC_MDIO lines.
14.6.1.4, 14-108	In Figure 14-113, superscript the ‘1’ on the EC_MDC and ECC_MDIO lines.
14.6.1.5, 14-109	In Figure 14-114, superscript the ‘1’ on the EC_MDC and ECC_MDIO lines.
15.4.1.3, 15-30	Add the following paragraph after the bulleted list: Note that when operating the DMA in chaining mode, the register byte count field, BCR[BC], must be initialized to zero before enabling the pause feature. In chaining modes, the channel does not pause for descriptor fetch transfer.

15.4.1.3, 15-30 Add the following sentence after the second sentence of the third paragraph:

Note, however, that write data for a paused transfer may not have reached the target interface when so indicated.

15.5, 15-39 Replace Figure 15-25 with the following:



16.3.1.3, 16-15 Add the following note after the first paragraph:

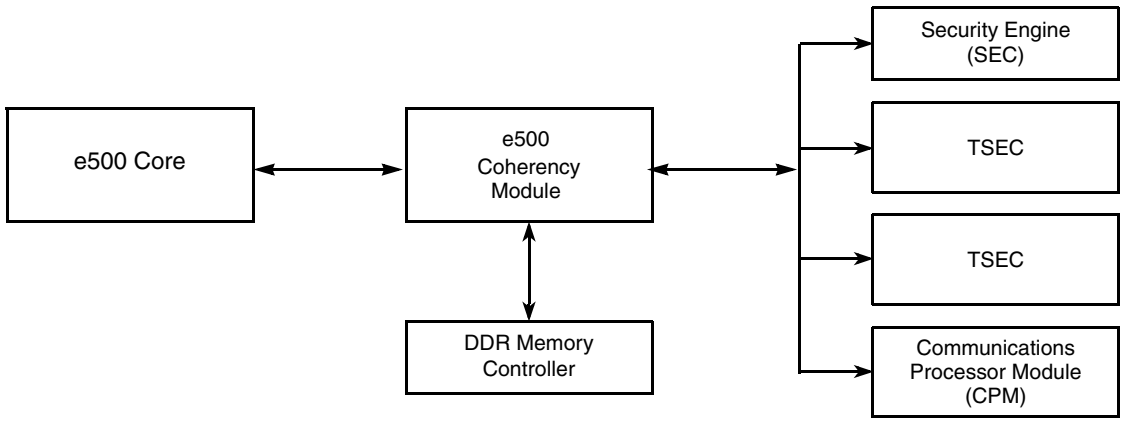
NOTE

PCI1 and PCI2 can also be used as possible target interfaces for PCI inbound ATMUs, mapping from one PCI to the other. However, it is illegal for PCI1 to use PCI1 as a target, or for PCI2 to use PCI2 for a target.

16.3.1.2.4, 16-25 In Figure 16-9, remove S_D from bit 1. This bit is reserved and should have a value of ‘0’.

16.3.1.3.4, 16-30	In Table 16-14, add the following targets to the PIWAR*n*[TGI] field description: 0000 PCI1 0001 PCI2
16.3.2.3, 16-42	In Table 16-26, modified the description for bit 2 to read: Indicates whether this PCI controller is configured as a master. For PCI1, the reset state of this bit depends on whether PCI1 is configured in host mode (bus master = 1) or agent mode (bus master = 0). 0 Disables the ability to generate PCI accesses 1 Enables this PCI controller to behave as a PCI bus master Note that the bus master bit in the PCI bus command register should be set before attempting an outbound configuration access.
16.3.2.17, 16-53	The register described in this section should be called PCI bus minimum grant (MIN GNT) register; its single field is MINGNT. (The field description is unchanged.)
16.4.2.8.2, 16-68	Replace the following bulleted item: – The 16-clock latency timer has expired, and the first data phase has not begun. with: – The 32-clock latency timer has expired, and the first data phase has not begun.
16.4.2.11.3, 16-76	Replace the second to the last sentence of the first paragraph with the following: When the MPC8555E is in agent lock mode, it retries all externally-generated PCI/X configuration cycles until the ACL bit in the PCI bus function register (0x44) is cleared.
16.4.2.11.4, 16-77	In Table 16-50, correct the table heading from ‘AD*n* Used in IIDSEL’ to ‘AD*n* Used in IDSEL’.
16.4.2.11.4, 16-77	In Table 16-50, separate entries for device number 0 and devices 1–9. Add the following table footnote related to IDSEL used for device number 0: [1] No external configuration transaction takes place; rather, internal registers are accessed. In addition, insert the following table footnote related to IDSEL used for device numbers 1–9: [2] No IDSEL line asserted. Type0 configuration transaction is run, but ends with a master abort since no device responds. Finally, renumber previous footnote accordingly.
16.5.1.3, 16-85	Insert the following sentence after the first sentence: The purpose of this mode is to allow initial configuration on the post by the local processor before opening the port to be further configured by the external host.
16.5.3, 16-85	Remove this section, as there is no support for RapidIO on this device.

17.1, 17-3 Replace Figure 17-1 with the following:



17.2, 17-11 In Table 17-2, the 'Offset' values for DEU, AESU, and MDEU modules show incorrect ranges. Replace the rows with the following:

0x3_2000–0x3_2FFF	DEU	DES/3DES execution unit	Crypto EU	17.4.2/17-33
0x3_4000–0x3_4FFF	AESU	AES execution unit		17.4.6/17-67
0x3_6000–0x3_6FFF	MDEU	Message digest execution unit		17.4.4/17-51

17.4.2.6, 17-43 In Table 17-19, replace the row for bit 61 with the following:

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 DEU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the DEU cannot accept FIFO inputs larger than 512 bytes without overflowing.

17.4.2.7, 17-45 In Table 17-20, replace the row for bit 61 with the following:

61	IFO	Input FIFO overflow error. 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 DEU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the DEU cannot accept FIFO inputs larger than 512 bytes without overflowing.

17.4.3.2.1, 17-48 In Table 17-21, replace the row for bits 53–55 with the following:

53–55	BURST SIZE	The AFEU implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/context. The AFEU signals to the channel that BURST SIZE amount of data is available to be pushed to or pulled from the FIFO. **Note:** The inclusion of this field in the AFEUMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the AFEU.

17.4.4.1, 17-57 In Table 17-26, replace the row for bits 53–55 with the following:

53–55	BURST SIZE	The MDEU implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/context. The MDEU signals to the channel that a BURST SIZE amount of data is available to be pushed to the FIFO. **Note:** The inclusion of this field in the MDEUMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the MDEU.

17.4.5.1, 17-67 Replace the fourth sentence in the first paragraph, with the following:

RNGMR also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the RNG as an initiator.

17.4.5.1, 17-68 In Table 17-33, replace the row for bits 53–55 with the following:

53–55	BURST SIZE	The RNG 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 RNGMR is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the RNG.

17.4.5.4, 17-70 In Table 17-35, change bit 62 from ‘Reserved’ to ‘ID’ and replace the row with the following:

Bits	Name	Description
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 14.6.2.2, “Interrupt Status Register (ISR)”). 0 RNG is not signaling done 1 RNG is signaling done

17.4.6.6, 17-80 In Table 17-42, replace the row for bit 61 with the following:

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 AESU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the AESU cannot accept FIFO inputs larger than 512 bytes without overflowing.

17.5.1.2, 17-93–17-94 In Tables 17-47 and 17-48, replace the rows for 0x7 and 0x8 with the following:

0x7	request_bytes_data_trans
0x8	request_bytes_data_trans_done

17.5.1.2, 17-95 In Table 17-49, correct the following crypto-channel state value definitions:

0x16 Trans_request_read_multi_eu_in
0x18 Trans_request_read
0x1F Trans_request_write_snoopout
0x21 Trans_request_write

17.5.1.4, 17-98 — Replace the fourth paragraph and fifth paragraphs with the following:

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.

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 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.

17.6.2.1–17.6.2.3, 17-103–17-105 — In Figures 17-64, 17-65, and 17-66, change bit 47 to ‘Reserved’.

17.6.2.5, 17-108 — In Table 17-56, add the following warning text to the MCR[SWR] field description:

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.

18.4, 18-4 — In Table 18-3, replace the Offset 0xE_0000 row with the following:

0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn_n*1*nn*	18.4.1.1/18-4

18.4.1.1, 18-5 — In Table 18-1, change the read (R) value of bits 9, 23, and 25, to 1.

18.4.1.11, 18-15 — In Figure 18-11 and Table 18-14, correctly document bit field 11 as ‘Reserved’.

18.4.1.11, 18-16 — In Table 18-14, correct the last sentence of the asserted state description for DEVDISR[E500] to read as follows:

Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core, including the timer facilities.

18.4.1.15, 18-20 — In Figure 18-15 and Table 18-18, correct the system version register reset value to read:

0x8079_0011 (for MPC8555E) / 0x807A_0011 (for MPC8541E)
0x8071_0011 (for MPC8555) / 0x8072_0011 (for MPC8541)

18.5.1.6, 18-26 — Replace the second paragraph with the following:

These register fields and their functional relationship are shown in Section 6.10.1, “Hardware Implementation-Dependent Register 0 (HID0),” and Section 6.5.1, “Machine State Register (MSR).” The *e500 Reference Manual* has details on accessing these power management control bits.

18.5.1.7, 18-28 — In Figure 18-19, correct the input of the NOR gate which drives internal signal core_tben, formerly noted as being driven by ‘DEVDISR[TB]’, to be noted as being driven by ‘DEVDISR[TB] or [E500]’.

19.4.7, 19-22 — In Table 19-10, insert a centered header row labeled ‘TSEC1 DMA Events’ directly between rows for events ‘Receive FIFO above 3/4’ and ‘DMA reads’.

20.4.1, 20-27 — In Table 20-26, add target IDs for ‘PCI2’ (with a hex value of 01) and ‘Security’ (with a hex value of 07).

21.1.1, 21-7–21-8 — In Table 21-1, remove reserved registers between each FCCE*n* and FCCM*n* at offsets 0x9_1312 and 0x9_1332.

21.1.1, 21-11–21-12 — In Table 21-1, add reserved registers between each SCCE*n* and SCCM*n* at offsets 0x9_1A12, 0x9_1A52, and 0x9_1A72.

21.2.9, 21-24 — In Table 21-2, replace the row for ‘RAM Base + 0x8AF0’ with the following:

RAM Base + 0x8AF0	REV_NUM	Hword	Microcode revision number 0x00E8

21.3.1, 21-25–21-26 — In Figure 21-9, change bits 18–25 from ‘—’ to ‘MCN,’ change bits 26–27 from ‘—’ to ‘EP.’

In Table 21-9, replace the rows for bits 18–25 and 26–27 with the following:

18–25	MCN	In FCC protocols, this field contains the protocol code as follows: 0x00 HDLC 0x0A ATM 0x0C Ethernet 0x0F Transparent
26–27	EP	Endpoint: Logical pipe number (only in USB) 00 Endpoint 0 01 Endpoint 1 10 Endpoint 2 11 Endpoint 3

21.4, 21-29 — Replace the first bullet with the following:

- 4 Kbytes of instruction RAM to store a microcode package of up to 1K instructions

21.4, 21-31 — In Figure 21-12, replace BD/Data in Banks #5 and #6 with Parameter RAM.

21.4.2, 21-33 — In Table 21-13, change the size of pages 13–16 (address 0x8C00) from 1280 to 1024.

24.1, 24-3 After the features list, add the following note:

NOTE

In FCC2 Utopia Master or Slave mode only, Clk13, Clk14, BRG 5, BRG6, and BRG7 are available.

30.13, 30-19–30-23 In Figures 30-10, 30-11, 30-14, and 30-16, change power supply voltage from 5 V to 3.3 V.

35.5.1, 35-10 Replace the two paragraphs following Figure 35-5 and replace Table 35-3 with the following sections and table:

35.5.1.1 Packet-Level Interface

If USEP1[RTE] is 0, the USB host controller uses a packet-level interface to communicate with the user. Each transmit packet is prepared in a buffer and referenced by a TxBD as described in Section 35.6.3, “USB Transmit Buffer Descriptor (Tx BD) for Host.” Each receive packet is stored in a buffer referenced by a RxBD as described in Section 35.6.1, “USB Receive Buffer Descriptor (Rx BD) for Host and Function.” A SETUP or OUT transaction requires at least two TxBDs, one for the token and one or more for the data packet. An IN transaction requires one TxBD for the token and one or more RxBDs for the data packet. Tokens are not checked for validity and are transmitted as is. The user is responsible for token validity as well as CRC5 generation.

35.5.1.2 Transaction-Level Interface

If USEP1[RTE] is 1, the USB host controller uses a transaction-level interface to communicate with the user. Each transaction uses one TrDB as described in Section 35.6.4, “USB Transaction Buffer Descriptor (TrBD) for Host.” The USB host controller generates the token based on the TOK field in the TrBD. For SETUP and OUT transactions, the TrBD points to a single buffer containing the data packet to be transmitted. For IN transactions, the TrBD points to a single buffer which is used for the receive data packet.

Table 35-3. USB Tokens

<table>
<tr><th>Token</th><th colspan="2">Description</th></tr>
<tr><td>OUT</td><td>Packet-Level Interface

Transmission begins when the USB host controller fetches a TxBD containing an OUT token and a data TxBD and loads them to the host FIFO. The token and data are transmitted and a handshake is expected.
If a handshake is not received within the expected time interval, the USB controller clears TxBD[R] of data BD, sets the TxBD[TO] indication and generates a TXE1 interrupt.
When STALL or NAK is received within the expected time interval, the USB controller clears TxBD[R] of data BD, sets the TxBD[STALL] or TXBD[NAK] indication and generates a TXE1 interrupt. When ACK is received within the expected time interval, the USB controller clears TxBD[R] of data BD, and generates an interrupt if TxBD[I] = 1. No indication is set.
The token TxBD[R] is cleared right after the OUT token transmission.</td><td>Transaction-Level Interface

Transmission begins when the USB host controller fetches a TrBD with the TOK field indicating an OUT transaction. The token is generated and then the data packet from the buffer is transmitted and a handshake is expected.
If a handshake is not received within the expected time interval, the USB controller clears TrBD[R], sets the TrBD[TO] indication and generates a TXE1 interrupt.
When STALL or NAK is received within the expected time interval, the USB controller clears TrBD[R], sets the TrBD[STALL] or TrBD[NAK] indication and generates a TXE1 interrupt. When ACK is received within the expected time interval, the USB controller clears TrBD[R], and generates a TXB interrupt if TrBD[I] = 1.No indication is set.</td></tr>
<tr><td></td><td colspan="2">USB Out Transaction
<table>
<tr><th>Token</th><th>Data</th><th>Handshake Received by Host</th><th>Indication on TxBD/TrBD</th></tr>
<tr><td rowspan="4">OUT</td><td rowspan="4">Sent by host</td><td>None</td><td>TO</td></tr>
<tr><td>ACK</td><td>None</td></tr>
<tr><td>NAK</td><td>NAK</td></tr>
<tr><td>STALL</td><td>STALL</td></tr>
</table></td></tr>
</table>

Table 35-3. USB Tokens (continued)

<table>
<tr><th>Token</th><th colspan="2">Description</th></tr>
<tr><td>IN</td><td>Packet-Level Interface

Transmission begins when the USB host controller fetches a TxBD containing an IN token and loads the token to FIFO. After the IN token is transmitted the USB host controller waits for reception of data within expected time interval. On reception of a valid DATA PID an RxBD is fetched. The received data and DATA PID are stored in receive FIFO. If RxBD[E] is set PID and data will be moved to the buffer. While receiving the data the USB host controller calculates CRC16, performs bit un-stuffing. On end of reception calculated CRC is compared to received and octet alignment is checked, RxBD[E] is cleared, RxBD[PID] is set according to received DATA PID and error indications are set if required: RxBD[CR] for failed CRC check, RxBD[NO] for non-octet sized data and RxBD[AB] if bit stuffing error occurred.
If no valid DATA PID or no data at all received during the expected time interval a TO indication in the token TxBD is set.</td><td>Transaction-Level Interface

Transmission begins when the USB host controller fetches a TrBD with the TOK field indicating an IN transaction. After the IN token is generated and transmitted, the USB host controller waits for reception of data within the expected time interval. The received data packet is stored in buffer reference by the TrBD. While receiving the data the USB host controller calculates CRC16 and performs bit un-stuffing. At end of the packet, the calculated CRC is compared to the received value and octet alignment is checked, TrBD[R] is cleared, TrBD[PID] is set according to the received DATA PID and error indications are set if required: TrBD[CR] for failed CRC check, TrBD[NO] for non-octet sized data and TrBD[AB] if bit stuffing error occurred. If any of the above errors are reported, TrBD[RXER] is also set, and a TXE1 interrupt is generated.
If no valid DATA PID or no data at all received during the expected time interval, a TrBD[TO] is set and a TXE1 interrupt is generated.
If no errors occurred and TrBD[I] is set, a TXB interrupt is generated to indicate successful completion of the transaction.</td></tr>
<tr><td></td><td colspan="2">USB In Transaction
<table>
<tr><th>Token</th><th>Data Transmitted by Function</th><th>Handshake Generated by Host</th><th>Indication on BD</th></tr>
<tr><td rowspan="3">IN</td><td>Received correctly</td><td>ACK</td><td>RxBD[E]/TrBD[R] is cleared</td></tr>
<tr><td>Received corrupted</td><td>None</td><td>RxBD[CR]/TrBD[CR} or RxBD[AB]/TrBD[AB] or RxBD[NO]/TrBD[NO]</td></tr>
<tr><td>None</td><td>None</td><td>TxBD[TO]/TrBD[TO]</td></tr>
</table></td></tr>
<tr><td>SETUP</td><td colspan="2">The format of setup transactions is similar to OUT but uses a SETUP rather than an OUT PID. A SETUP token is recognized only by a control endpoint. When a SETUP token is received, setup reception begins. The USB controller fetches the next BD associated with the endpoint; if it is empty, the controller starts transferring the incoming packet to the buffer. When the buffer is full, the USB controller clears RxBD[E] and generates an interrupt if RxBD[I] = 1. If the incoming packet is larger than the buffer, the USB controller fetches the next BD and, if it is empty, continues transferring the rest of the packet to this buffer. The entire data packet including the DATA0 PID is written to the receive buffers. If the packet was received without CRC or bit stuff errors, an ACK handshake is sent to the host. If an error occurs, no handshake packet is returned and error status bits are set in the last RxBD associated with this packet.</td></tr>
<tr><td>Start of Frame (SOF)</td><td colspan="2">When an SOF packet is received, the USB controller issues a SOF maskable interrupt and the frame number entry in the parameter RAM is updated.</td></tr>
<tr><td>Preamble (PRE)</td><td colspan="2">The PRE token signals the hub that a low-speed transaction is about to occur. The PRE token is read only by the hub. The USB controller ignores the PRE token function in function mode.</td></tr>
</table>

35.5.2, 35-12 Replace Section 35.5.2 with the following:

35.5.2 SOF Transmission for USB Host Controller

The following section describes the mechanism used by the USB host controller to support the automatic transmission of SOF tokens. This mechanism is enabled by setting USMOD[SFTE].

SOF packets should be transmitted every 1 ms. Since the time interval between two SOF packets must be more precise than could be accomplished by software, a hardware timer is used to assert an interrupt to the CP. When the interrupt is serviced by the CP, it prepares a SOF token and loads it to the host endpoint. Once the SOF token is loaded to the FIFO, it is transmitted like any other packet.

Before each SOF transmission, the software should prepare a value for the frame number and CRC5 to be transmitted in SOF token and place it in the parameter RAM (for further details please refer to Section 35.5.5, “Frame Number (FRAME_N).” One possible implementation would be to use the SOF interrupt (see Section 35.5.7.5, “USB Event Register (USBER)”) to prepare the frame number for the next SOF packet. The SFT interrupt should not be used for this purpose since it is generated before the SOF packet is actually transmitted.

The application software should also guarantee that the USB host has completed all pending transactions prior to the 1 ms tick, so that the transmit FIFO is empty at this point. The current value of the SOF timer may be read at any time to synchronize the software with the USB frames. See Section 35.5.7.8, “USB Start of Frame Timer (USSFT)” for more information.

35.5.4, 35-15 In Table 35-5, replace the row for offset 0x1E with the following:

0x1E	**HIMMR**	16 bits	When using the transaction-based interface in host mode, this field must be programmed to {CCSRBAR[0:11], 0x9}. Otherwise, this field is unused.

35.5.6, 35-17 Table 35-8, replace the row for bit 6 with the following:

6	DTB	Data bus indicator 0 Use system bus for SDMA operation 1 Use local bus for SDMA operation

35.5.7.1, 35-17 In Figure 35-11, change bit 4 from ‘Reserved’ to ‘SFTE’.

In Table 35-9, replace the row for bits 2–4 with the following:

2–3	—	Reserved, should be cleared.
4	SFTE	Start-of-frame timer enable. Setting this bit enables the start-of-frame timer and automatic SOF transmission. See Section 35.5.7.8, “USB Start of Frame Timer (USSFT),” and Section 35.5.2, “SOF Transmission for USB Host Controller,” for more information. 0 SOF timer is disabled 1 SOF timer is enabled **Note:** When SFTE is 1, the PC21 pin cannot be used as CP_INT since the CP interrupt is used internally for generating the SOF packet.

35.5.7.4, 35-20 In Figure 35-14, change bits 2 and 3 from ‘Reserved’ to ‘ISFT’ and ‘DSFT’, respectively.

In Table 35-12, replace the row for bits 2–5 with the following:

2	ISFT	Increment start-of-frame time. Setting the ISFT bit increments the start-of-frame time by one. This bit could be used to synchronize the USB frames to an external timing source.
3	DSFT	Decrement start-of-frame time. Setting the DSFT bit decrements the start-of-frame time by one. This bit could be used to synchronize the USB frames to an external timing source.
4–5	—	Reserved, should be cleared.

35.5.7.5, 35-21 In Figure 35-15, change bit 5 from ‘Reserved’ to ‘SFT’.

In Table 35-13, replace the row for bits 0–5 with the following:

0–4	—	Reserved, should be cleared.
5	SFT	The start-of-frame timer (USSFT[SFT]) wrapped from 11,999 to 0.

35.5.7.8, 35-22 Insert the following section after Section 35.5.7.7, and subsequently, renumber the tables and figures that follow:

35.5.7.8 USB Start of Frame Timer (USSFT)

When enabled by USMOD[SFTE], the USSFT contains the current time within the frame with a resolution of one bit time. When the value of USSFT wraps from 11,999 to 0, a CP interrupt is asserted to trigger the transmission of a SOF packet, and USBER[SFT] is set.

The USSFT may be read at any time.

	0	1	2 … 15
Field	—		SFT
Reset	0010_1110_1101_1000		
R/W	R		
Addr	0x_1B78		

Figure 35-17. USB Start of Frame Timer (USSFT)

Table 35-15 describes USSFT fields.

Table 35-15. USSFT Fields

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–15	SFT	Start-of-frame time. This field contains the number of bit times since the last SOF trigger. Note that the actual SOF transmission occurs slightly later.

35.6.1, 35-24 Replace the paragraph above Figure 35-18 with the following:

The RxBD is identical for both the host mode (when using the packet-level interface) and the function mode. There are no RxBDs in host mode when using the transaction-level interface.

35.6.3, 35-28 Before the first paragraph, add the following paragraph:

The TxBD described in this section is used when the packet-level interface is active. See Section 35.5.1.1, “Packet-Level Interface,” for more information.

35.6.3, 35-30 After Section 35.6.3, add the following section:

35.6.4 USB Transaction Buffer Descriptor (TrBD) for Host

The TrBD described in this section is used when the transaction-level interface is active. See Section 35.5.1.2, “Transaction-Level Interface,” for more information.

Data to be transmitted with the USB to the CP by is arranged in buffers referenced by the TrBD ring. The first word of the TrBD contains status and control bits.

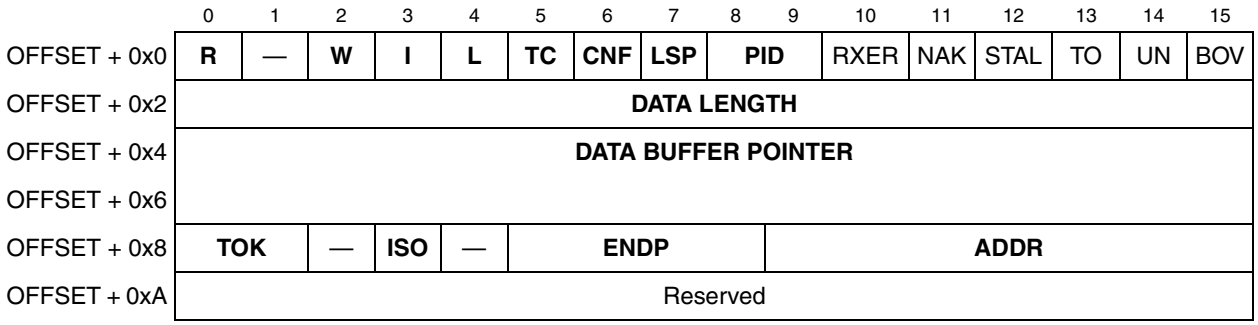


Figure 35-20. USB Transaction Buffer Descriptor (TrBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 35-18 describes USB TrBD fields.

Table 35-18. USB Host TrBD Fields

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared.

Table 35-18. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TrBD table. 1 This is the last BD in the TrBD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASE). The number of TrBDs in this table is programmable, and is determined only by the TrBD[W] and the overall space constraints of the dual-port RAM.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 The TXB bit in the event register is set when this buffer is serviced. TXB can cause an interrupt if it is enabled.
	4	**L**	Last This bit should always be 1 since each TrBD represents an entire transaction.
	5	**TC**	Transmit CRC. Append CRC to transmitted data packet. 0 Transmit end-of-packet after the last data byte. This setting can be used for testing purposes to send a bad CRC after the data. 1 Transmit the CRC sequence after the last data byte.
	6	**CNF**	Transmit confirmation. This bit should always be set to 1 to obtain confirmation for each transaction.
	7	**LSP**	Low-speed transaction. 0 This transaction is with the host or a full-speed device. 1 This transaction is with a low-speed device. Transmit a PRE packet before the token.
	8–9	**PID**	Packet ID. For OUT/SETUP transactions, this field is prepared by the user with the following values: 0X Do not append PID to the data packet. 10 Transmit DATA0 PID before sending the data packet. 11 Transmit DATA1 PID before sending the data packet. For IN transactions, this field is provided by the USB host controller with the following values: 00 Buffer contains DATA0 packet. 01 Buffer contains DATA1 packet.
	10	RXER[1]	Receive error. This bit indicates that an error was detected while receiving the data packet of an IN transaction. If RXER is 1, bits 11–15 have a different meaning as explained below.
	11	NAK/NO[1]	RXER = 0: NAK received. Indicates that the endpoint has responded with a NAK handshake (OUT transaction). The packet was received error-free; however, the endpoint could not accept it. RXER = 1: Rx non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data has been placed into the associated data buffer.
	12	STAL/AB[1]	RXER = 0: STALL received. Indicates that the endpoint has responded with a STALL handshake (OUT transaction). The endpoint needs attention through the control pipe. RXER = 1: Frame aborted. Bit stuff error occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.

Table 35-18. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
	13	TO/CR[1]	RXER = 0: Time out. Indicates that the endpoint failed to acknowledge the token (IN transaction) or the data packet (OUT/SETUP transaction). RXER = 1: CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer. Written by the USB controller after the received data has been placed into the associated data buffer.
	14	UN/OV[1]	RXER = 0: Underrun. Indicates that the USB encountered a transmit FIFO underrun condition while sending the data packet (OUT/SETUP transaction). RXER = 1: Overrun. An internal receive FIFO overrun occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	15	BOV[1]	Buffer overflow. IN transactions only. Indicates that the number of received bytes is larger than the buffer size as provided in the data length field.
0x02	0–15	**Data Length**	For OUT/SETUP transactions, the user prepares this field with the number of bytes to be sent from the data buffer. It will not be modified by the CP. For IN transactions, the user prepares this field with the size of the data buffer, which must be divisible by 4. The CP will return the actual number of bytes written to the data buffer. If the number of received bytes, including the 2-byte CRC, is larger than the data buffer, the BOV bit will be set by the CP.
0x04	0–31	**Data Buffer Pointer**	The data buffer pointer. The buffer may reside in either internal or external memory. For OUT/SETUP transactions, this points to the buffer containing the data packet to transmit. It may have any alignment. For IN transactions, this points to the buffer into which the data packet should be received, The pointer must be divisible by 4.
0x08	0–1	**TOK**	Token type This field determines the type of token to be transmitted and the type of transaction. 00 SETUP 01 OUT 10 IN 11 Reserved
	2	—	Reserved, should be cleared.
	3	**ISO**	Isochronous This bit indicates that the transaction is isochronous, so no handshake is required. 0 Bulk/control/interrupt. The handshake packet is automatically expected or generated by the USB host controller. 1 Isochronous. No handshake packets are expected or generated. This bit actually controls the value that is written to USEP1[TM] before processing this transaction.
	5	—	Reserved, should be cleared.
	5–8	**ENDP**	Endpoint This field indicates the endpoint number to be included in the token.
	9–15	**ADDR**	Address This field indicates the device address to be included in the token.
0x0A	0–15	—	Reserved, should be cleared.

[1] Written by the USB controller after it finishes sending or receiving the associated data buffer.

35.8, 35-31 Add the following row to the end of Table 35-19:

Buffer Overflow	For **USB host mode** packet-level interface only. If the received data packet is larger than the allocated buffer, the remaining data is discarded, and TrBD[BOV] is set. The TXE1 interrupt bit is set.

35.9, 35-31 Insert the following sections after Section 35.8:

35.9 USB Function Controller Initialization Example

The following is an example initialization sequence for the USB controller operating in function mode. It can be used to setup two function endpoints to fill transmit FIFOs so that data is ready for transmission when an IN token is received from the USB. The tokens can be generated using a USB traffic generator.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Clear FRAME_N.
4. Write (DPRAM+0x500) to EP1PTR, and (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
5. Write 0xBC80_0004 to DPRAM+0x20 to set up the TxBD[Status and Control, Data Length] fields of endpoint 1.
6. Write DPRAM+0x200 to DPRAM+0x24 to set up the TxBD[Buffer Pointer] field of endpoint 1.
7. Write 0xBCC0_0004 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of endpoint 2.
8. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of endpoint 2.
9. Write 0xCAFE_CAFE to DPRAM+0x200 to set up the endpoint 1 Tx data pattern.
10. Write 0xFACE_FACE to DPRAM+0x210 to set up the endpoint 2 Tx data pattern.
11. Write 0x0000_0020 to DPRAM+0x500 to set up the RBASE and TBASE fields of the endpoint 1 parameter RAM.
12. Write 0x1818_0100 to DPRAM+0x504 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 1 parameter RAM.
13. Write 0x0000_0020 to DPRAM+0x508 to set up the RBPTR and TBPTR fields of the endpoint 1 parameter RAM.
14. Clear the TSTATE field of the endpoint 1 parameter RAM.
15. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
16. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
17. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.

18. Clear the TSTATE field of the endpoint 2 parameter RAM.
19. Write 0x0000 to USEP1: Endpoint Number 0, control transfer, one packet only, and normal handshake.
20. Write 0x7200 to USEP2: Endpoint Number 7, bulk transfer, one packet only, and normal handshake.
21. Write 0x00 to the USMOD for full-speed 12 Mbps function endpoint mode and disable the USB.
22. Write 0x05 to the USAD for slave address 5.
23. Set USMOD[EN] to enable the USB controller.
24. Write 0x80 to USCOM to start filling the Tx FIFO with endpoint 1 data ready for transmission when an IN token is received.
25. Write 0x81 to USCOM to start filling the Tx FIFO with endpoint 2 data ready for transmission when an IN token is received.
26. Generate an IN token to address 5, endpoint number 0, control.
27. Generate an IN token to address 5, endpoint number 7, bulk.

35.10 Programming the USB Host Controller (Packet-Level)

The MPC8555E implementation of a USB host uses the endpoint represented by USEP1 to control the host transmission and reception. The other endpoints are typically not used, except for testing purposes (loopback).

Programming the USB controller to act as host is similar to configuring an endpoint for function operation. A general outline of how to program the host controller follows. (A more detailed example can be found in Section 35.10.1, “USB Host Controller Initialization Example.”)

- Set the host bit in the mode register (USBMOD[HOST] = 1) to configure the controller as a host.
- Set the multi-frame bit in the endpoint configuration register (USEP1[MF] = 1) to allow SETUP/OUT tokens and DATA0/DATA1 packets to be sent back-to-back.
- Prepare tokens in separate BDs.
- Using software, append the CRC5 as part of the transmitted data because the CPM does not support automatic CRC5 generation.
- Clock the USB host controller as a high speed function (48-MHz reference clock).
- For low-speed transactions with an external hub, set TxBD[LSP] in the token’s BD. This causes the USB host controller to generate a preamble (PRE token) at full speed before changing the transmit rate to low speed and sending the data packet. After completion of the transaction, the host returns to full-speed operation. Note that LSP should be set only for token BDs.

35.10.1 USB Host Controller Initialization Example

The following is a local loopback example initialization sequence for the USB controller operating as a host. It can be used to set up the host endpoint and one function endpoint to demonstrate an IN token transaction.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Write (DPRAM+0x500) to EP1PTR, (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
4. Write 0x0000_0020 to DPRAM+0x500 to set up the RBASE and TBASE fields of the host endpoint parameter RAM.
5. Write 0x1818_0100 to DPRAM+0x504 to set up the RFCR, TFCR, and MRBLR fields of the host endpoint parameter RAM.
6. Write 0x0000_0020 to DPRAM+0x508 to set up the RBPTR and TBPTR fields of the host endpoint parameter RAM.
7. Clear the TSTATE field of the host endpoint parameter RAM.
8. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
9. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
10. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
11. Clear the TSTATE field of the endpoint 2 parameter RAM.
12. Write 0xB000_0000 to DPRAM+0x00 to set up the RxBD[Status and Control, Data Length] fields of the host endpoint.
13. Write DPRAM+0x100 to DPRAM+0x04 to set up the RxBD[Buffer Pointer] field of the host endpoint.
14. Write 0xB800_0003 to DPRAM+0x20 to set up the TxBD[Status and Control, Data Length] fields of the host endpoint.
15. Write DPRAM+0x200 to DPRAM+0x24 to set up the TxBD[Buffer Pointer] field of the host endpoint.
16. Write 0xBC80_0003 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of the function endpoint.
17. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of the function endpoint.
18. Write 0x698560 to DPRAM+0x200 to set up the host endpoint Tx data pattern. This pattern consists of the IN token and the CRC5.
19. Write 0xABCD_1234 to DPRAM+0x210 to set up the function endpoint Tx data pattern.
20. Write 0x0020 to USEP1 for the host: non-isochronous transfer, multi-packet, packet-level interface.

21. Write 0x1100 to USEP2 for the function: interrupt transfer, one packet only.
22. Write 0x06 to USMOD for full-speed 12 Mbps signaling, local loopback configuration (test and host modes set), and disable the USB.
23. Write 0x05 to the USAD for slave address 5.
24. Set USMOD[EN] to enable the USB controller.
25. Write 0x81 to the USCOM to start filling the Tx FIFO with endpoint 2 data to be ready for transmission when an IN token is received.
26. Write 0x80 to the USCOM to start transmitting the IN token.

The expected results are as follows:

- TxBD[Status and Control] of the host endpoint should contain 0x3800.
- TxBD[Data Length] of the host endpoint should contain 0x0003.
- TxBD[Status and Control] of endpoint 2 should contain 0x3C80.
- TxBD[Data Length] of endpoint 2 should contain 0x0003.
- RxBD[Status and Control] of the host endpoint should contain 0x3C00.
- RxBD[Data Length] of the host endpoint should contain 0x0005.
- The receive buffer of the host endpoint should contain 0xABCD_122B, 0x42xx_xxxx.

35.11 Programming the USB Host Controller (Transaction-Level)

The MPC8555E implementation of a USB host uses the endpoint represented by USEP1 to control the host transmission and reception. The other endpoints are typically not used, except for testing purposes (loopback).

Programming the USB controller to act as host is similar to configuring an endpoint for function operation. A general outline of how to program the host controller follows. (A more detailed example can be found in Section 35.11.1, “USB Host Controller Initialization Example.”)

- Set the host bit in the mode register (USBMOD[HOST] = 1) to configure the controller as a host.
- Set the multi-frame bit in the endpoint configuration register (USEP1[MF] = 1) to allow SETUP/OUT tokens and DATA0/DATA1 packets to be sent back-to-back.
- Set USEP1[RTE] to enable the transaction-level interface.
- Clock the USB host controller as a high speed function (48-MHz reference clock).
- For low-speed transactions with an external hub, set TrBD[LSP]. This causes the USB host controller to generate a preamble (PRE token) at full speed before changing the transmit rate to low speed and sending the token. After completion of the transaction, the host returns to full-speed operation.

35.11.1 USB Host Controller Initialization Example

The following is a local loopback example initialization sequence for the USB controller operating as a host. It can be used to set up the host endpoint and one function endpoint to demonstrate an IN token transaction.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Write (DPRAM+0x500) to EP1PTR, (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
4. Write 0x0020 to DPRAM+0x502 to set up the TBASE field of the host endpoint parameter RAM.
5. Write 0x1818 to DPRAM+0x504 to set up the RFCR and TFCR fields of the host endpoint parameter RAM.
6. Write 0x0020 to DPRAM+0x50a to set up the TBPTR field of the host endpoint parameter RAM.
7. Clear the TSTATE field of the host endpoint parameter RAM.
8. Initialize the HIMMR field of the host endpoint parameter RAM.
9. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
10. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
11. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
12. Clear the TSTATE field of the endpoint 2 parameter RAM.
13. Write 0xB800_0040 to DPRAM+0x20 to set up the TrBD[Status and Control, Data Length] fields of the host endpoint.
14. Write DPRAM+0x100 to DPRAM+0x24 to set up the TrBD[Buffer Pointer] field of the host endpoint.
15. Write 0x8085 to DPRAM+0x28 to set up the TrBD token fields of the host endpoint.
16. Write 0xBC80_0003 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of the function endpoint.
17. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of the function endpoint.
18. Write 0xABCD_1234 to DPRAM+0x210 to set up the function endpoint Tx data pattern.
19. Write 0x0030 to USEP1 for the host: non-isochronous transfer, multi-packet, transaction-level interface.
20. Write 0x1100 to USEP2 for the function: interrupt transfer, one packet only.
21. Write 0x06 to USMOD for full-speed 12 Mbps signaling, local loopback configuration (test and host modes set), and disable the USB.
22. Write 0x05 to the USAD for slave address 5.

23. Set USMOD[EN] to enable the USB controller.
24. Write 0x81 to the USCOM to start filling the Tx FIFO with endpoint 2 data to be ready for transmission when an IN token is received.
25. Write 0x80 to the USCOM to start transmitting the IN token.

The expected results are as follows:

- TrBD[Status and Control] of the host endpoint should contain 0x3800.
- TrBD[Data Length] of the host endpoint should contain 0x0005.
- TxBD[Status and Control] of endpoint 2 should contain 0x3C80.
- TxBD[Data Length] of endpoint 2 should contain 0x0003.
- The receive buffer of the host endpoint should contain 0xABCD_122B, 0x42xx_xxxx.

37.8, 37-12 In Table 37-4, replace the rows for offset 0x00 and 0x02 with the following:

0x00	RIPTR	Hword	Receive internal temporary data pointer. Used by microcode as a temporary buffer for data. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR. Must be 32-byte aligned and the size of the internal buffer must be 32 bytes, unless it is stated otherwise in the protocol specification.
0x02	TIPTR	Hword	Transmit internal temporary data pointer. Used by microcode as a temporary buffer for data. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR. Must be 32-byte aligned and the size of the internal buffer must be 32 bytes, unless it is stated otherwise in the protocol specification.

41.1, 41-1 Change the first bullet to read as follows:

Full duplex segmentation and reassembly at 155 Mbps

41.10.1, 41-38 In Table 41-11, replace the rows for offsets 0x40, 0x42, and 0x44 with the following:

0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.

41.10.l.3, 41-41 Figure 41-24, change bit 2 to read GLB.

41.10.l.3, 41-41 Table 41-13, replace the rows for bits 0–5, 6, and 7 with the following:

0–1	—	Reserved, should be cleared.
2	—	Global. Asserting GBL enables snooping of connection tables. GBL should not be asserted if any of the related DMAs will access the local bus.
3–5	—	Reserved, should be cleared.
6	ALB	Address look up bus for CAM or address compression tables 0 Reside on the system bus. 1 Reside on the local bus.
7	CTB	External connection tables bus 0 Reside on the system bus. 1 Reside on the local bus

41.10.2.2, 41-45 Table 41-15, replace the rows for bits 6 and 7 with the following:

0x00	6	DTB	Data buffers bus 0 Data buffers reside on the system bus. 1 Data buffers reside on the local bus
	7	BIB	BD, interrupt queues, free buffer pool and external SRTS logic bus 0 Reside on the system bus. 1 Reside on the local bus.

41.10.2.3, 41-52 Table 41-20, replace the rows for bits 6 and 7 with the following:

0x00	6	DTB	Data buffer bus 0 Reside on the system bus. 1 Reside on the local bus.
	7	BIB	BD, interrupt queue and external SRTS logic bus 0 Reside on the system bus. 1 Reside on the local bus. **Note:** When using AAL5, AAL1 CES in UDC mode, BDs and data should be placed on the same bus (TCT[DTB] = TCT[BIB]).

41.14, 41-95 Table 41-50, replace the row for bit 5 with the following:

0x86	5	CTB	Connection tables bus. Used for external channels only 0 External connection tables reside on the system bus. 1 External connection tables reside on the local bus.

42.5, 42-36 Table 42-13, replace the rows for offsets 0x40, 0x42, and 0x44 with the following:

0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 32 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User to specify only the lowest 16 bits of the DPRAM address offset calculated from the value in CCSRBAR.

45.5, 45-18 After the second paragraph in Section 45.5, add the following note:

NOTE

In the MPC8555E CPM PIO port pinmuxing, FCC2 only has a primary option available for FCC2_TxAddr0, FCC2_RxAddr0. There are no secondary options for these signals in the CPM PIO port pinmuxing.

B.2 Changes from Revision 0 to Revision 1

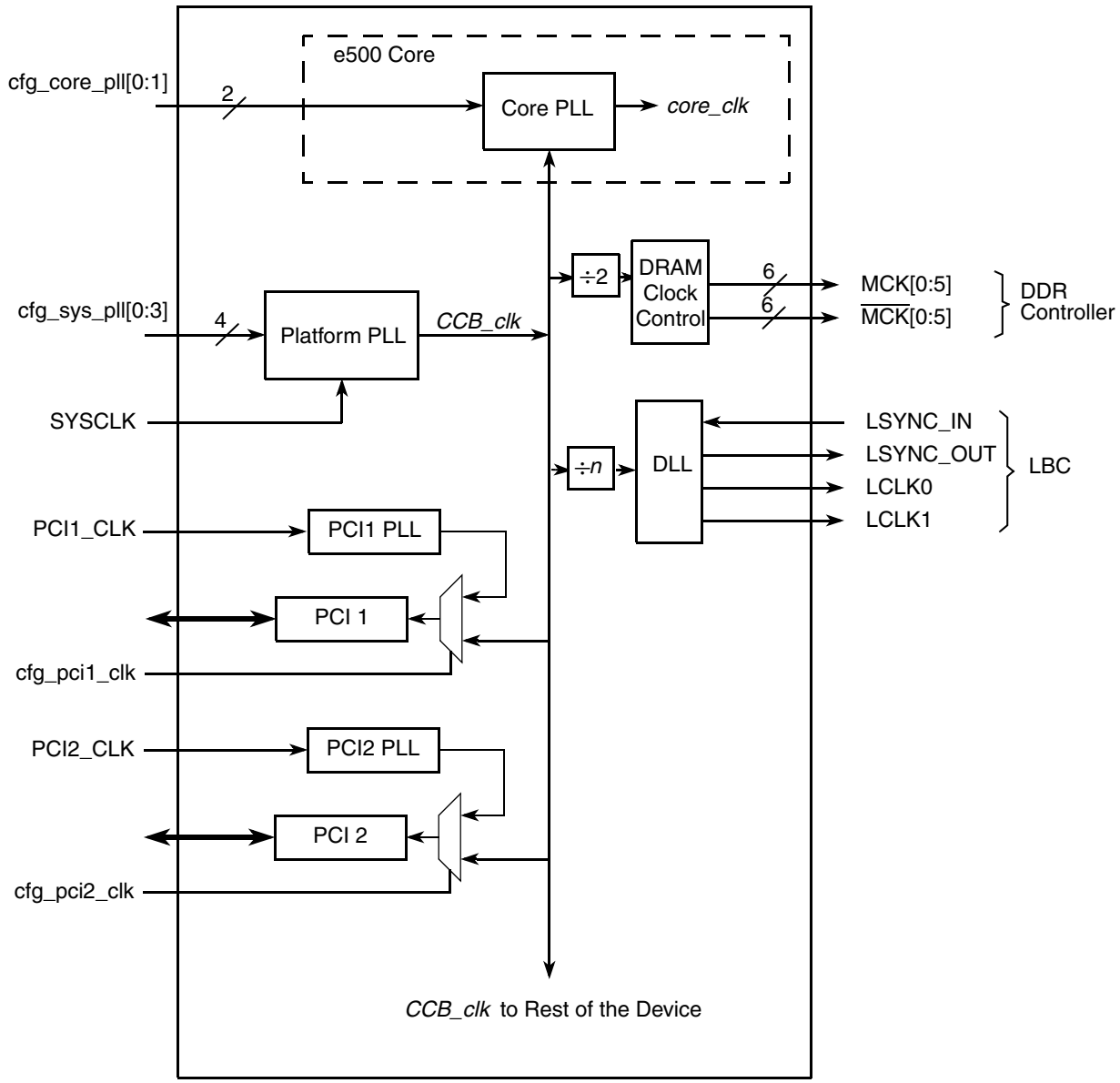
Major changes to the *MPC8555E PowerQUICC™ III Integrated Processor Reference Manual*, from Revision 0 to Revision 1 are as follows:

Section, Page	Changes
1.3.16, 1-23	Change the second paragraph to read as follows: "Six differential clock pairs are generated for DDR DRAMs. DLLs are used in the local bus memory controller (LBC) to generate two clock outputs."
2.4, 2-18	In Table 2-9, correct the reset value for the I2CDFSRR register to be 0x10.
2.4, 2-27	In Table 2-9, add the following rows under the TSEC1 FIFO Control and Status Registers section:

0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	14.5.3.2.1/14-30
0x2_4050– 0x2_4088	Reserved	R	0x0000_0000	—

2.4, 2-27	In Table 2-9, correct the reset value for the OSTBD register to be 0x0800_0000.
2.4, 2-28	In Table 2-9, correct the MACCFG1 register from having R/W, R access to R/W access.
2.4, 2-30	In Table 2-9, correct the CAR1 and CAR2 registers from having R access to R/W access.
2.4, 2-31	In Table 2-9, correct the ATTR and ATTRELI registers from having R access to R/W access.
2.4, 2-37	In Table 2-9, correct the offset for the Reserved addresses after the EOI register to be 0x4_00C0–0x4_0FF0.
2.4, 2-54	In Table 2-9, correct the reset value for the GPINDR register to be 0x*nnnn*_0000.
2.4, 2-55	In Table 2-9, remove DDRDLLCR designation at memory location 0xE_0E10.
2.4, 2-56	In Table 2-9, correct the name of the register with offset 0xE_1094 from PMLCB5 to PMLCB8.
3.1, 3-3 and 3-4	In Figure 3-1 and Table 3-1, remove memory interface signals, "MSYNC_IN," and "MSYNC_OUT."
3.1, 3-12	In Table 3-2, remove memory interface signals, "MSYNC_IN," and "MSYNC_OUT."
Chapters 4, 16, 18, 20	Throughout all applicable chapters, correctly state the debug mode source ID (formerly documented to be visible on signals PCI_AD[63:59]) to be visible on signals PCI_AD[62:58].

4.2.2, 4-3	In Table 4-3, correct the last sentence of the SYSCLK description to read as follows: "The CCB clock, in turn, feeds the PLL in the e500 core and the DLL that creates the local bus memory clocks."
4.4.2, 4-10	Correct the ninth step to read as follows: "9. The internal hard reset to the e500 core is negated and soft resets are negated to the DLL and other remaining I/O blocks. The DLL begins to lock."
4.4.3.3, 4-14	In Table 4-11, change the third functional signal used to define boot ROM location (formerly, "LWE_B[3]") to read "$\overline{\text{LWE}}$[3]".
4.4.4.1, 4-22	Replace the last sentence of the first paragraph with the following: "The CCB clock also feeds the PLL in the e500 core and the DLL that creates clocks for the local bus memory controller."
4.4.4.1, 4-23	Replace Figure 4-6, "Clock Subsystem Block Diagram," with the following figure:



4.4.4.3, 4-25 In Figure 4-7, replace the sentence:

"Watchdog timer events based on one of the 64 TB bits selected by TCR[WP] concatenated with the EIS-defined TCR[WPEXT] (WP||WPEXT)."

with the following sentence:

"Watchdog timer events based on one of the 64 TB bits selected by concatenating TCR[WPEXT] with the EIS-defined TCR[WP] (WPEXT||WP)."

4.4.4.3, 4-25 In Figure 4-7, replace the sentence:

"Fixed-interval timer events based on one of the 64 TB bits selected by TCR[FP] concatenated with the EIS-defined TCR[FPEXT] (FP||FPEXT)."

with the following sentence:

	"Fixed-interval timer events based on one of the 64 TB bits selected by concatenating TCR[FPEXT] with the EIS-defined TCR[FP] (FPEXT||FP)."
6.6.1, 6-14	In Table 6-8, change the description of TCR[WRC] as follows: 01 If MSR[ME] = 0, the second timeout is ignored. If MSR[ME] = 1, a machine check condition occurs on a second timeout of the watchdog timer, and if HID0[EMCP] = 1, the machine check interrupt is generated.
6.6.1, 6-14	In Table 6-8, change the second sentence of the description for bits 32–33 of the TCR register to the following: WPEXT[0–3] || WP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) WPEXT[0–3] || WP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
6.6.1, 6-15	In Table 6-8, change the second sentence of the description for bits 38–39 of the TCR register to the following: FPEXT[0–3] || FP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) FPEXT[0–3] || FP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
6.7.2.4, 6-22	In Table 6-12, change the bit range "43–46" to "36–46".

6.10.2, 6-27 In Table 6-17, replace the description of HID1[RFXE] with the following:

Read fault exception enable. Controls whether assertion of *core_fault_in* causes a machine check interrupt. The assertion of *core_fault_in* can result from an L2 multibit ECC error. It can also occur for a system error if logic on the integrated device signals a fault for nonfatal errors (read data is corrupt (or zero), but the bus transaction can complete without corrupting other system state, making it unnecessary to take a machine check (for example, a master abort of a PCI transaction). 0 Assertion of *core_fault_in* cannot cause a machine check. RFXE should be left clear if an interrupt is to be reported by the integrated device through *int* or *c_int* for this condition. If RFXE = 0, it is important that the integrated device generates an interrupt if *core_fault_in* is asserted. If *core_fault_in* is asserted, any data on the CCB is dropped, stalling the load/store unit and eventually causing the e500 pipeline either to stall until an interrupt occurs (typically generated by the programmable interrupt controller (PIC) in response to the fault) or to continue processing with bad data until the interrupt occurs. Because *core_fault_in* cannot cause a machine check, if RFXE is 0, it is critical that the system be configured to generate the appropriate interrupt. It is also possible to hang the processor (requiring a hard reset to recover) if a guarded load hits in the L2 cache and gets an uncorrectable ECC error. Because of this, avoid defining memory as cacheable but guarded. If this combination is required, RFXE must be enabled, in which case an error causes both a machine check interrupt and an external interrupt when a bus fault condition is detected unless interrupts are masked for all sources of bus faults, such as DRAM ECC errors, PCI parity errors, local bus parity errors, and others. RFXE must also be set if software requires that code execution stop immediately when a bus fault occurs rather than continuing with the bad data until the interrupt arrives. Again, this results in both a machine check interrupt and an external interrupt when a bus fault is detected, unless all possible sources for bus fault have their interrupts masked. The machine check interrupt can then re-enable normal interrupts and wait for the interrupt due to the fault to be received before returning from the machine check. 1 A machine check can occur due to assertion of *core_fault_in*. If MSR[ME] = 1 and a fault is signaled, a machine check interrupt occurs. If MSR[ME] = 0 and a fault is signaled, a checkstop occurs. Note that if RFXE is set and another mechanism is configured to generate an interrupt in response to assertion of *core_fault_in*, the same event causes two interrupts, the machine check enabled by setting RFXE and the interrupt triggered by the on-chip peripheral or other block; therefore, RFXE should be set only if no other mechanism is configured to generate an interrupt for this case. Note that the L2 cache detects any assertion of *core_fault_in* and ensures that the L2 cache is not corrupted when data is dropped for this type of transaction.

6.11.3, 6-30 In Table 6-20, describe bits 34–38 as “Reserved, should be cleared,” as is properly expressed in Figure 6-35. Disregard descriptions associated with bits 34, 35, or 36 in Table 6-20.

6.12.5.3, 6-37 In Table 6-28, describe bits 52–56 as “Reserved for implementation-specific use”.

7.3.1.1, 7-8 In Table 7-2, change the description for bit 1 from “Accesses to memory-mapped SRAM are unaffected...” to “Data in memory-mapped SRAM regions is unaffected...”

7.3.1.5.2, 7-16 In Table 7-12, change the bit descriptions for bits 27 and 31 of the L2ERRDET register as follows:
Bit 27, TPARERR: Add the following text: “Note that if an L2 cache tag parity error occurs on an attempt to write a new line, the L2 cache must be flash invalidated. L2 functionality is not guaranteed if flash invalidation is not performed after a tag parity error.”
Bit 31, L2CFGERR: Add the following text to the 1st line of description, “Reports

inconsistencies between the L2SIZ, L2BLKSZ, and L2SRAM settings of the L2 control register (L2CTL)."

7.7.4, 7-26 Add the following note after the second paragraph:

NOTE

There is a scenario in which a lock clear operation appears to fail to clear a lock in the L2 cache. This occurs only when the attempt to set the lock results in a bus error (for example, PCI returns an error condition).

Assume the following scenario:

1. The e500 attempts to set a lock in the L2 cache (by executing a **dcbtls** or **icbtls** instruction with CT = 1). The line is not already present in the cache, so it must be read from external memory. This read encounters an error which, depending on the chip configuration, will be reported to the core (probably as an interrupt).

2. At (or near) the same time, a cache external write to the same cache line is being mastered by the ECM.

3. Very soon after the cache external write, a transaction to clear the lock occurs. This can be caused by the processor executing a **dcblc** or **icblc** instruction with CT = 1, or by the ECM mastering a lock clear transaction.

If this scenario occurs within a tight timing window, the cache line may unexpectedly remain locked at the end of the sequence.

The interrupt handler may want to clear the erroneously remaining lock in this case.

7.9, 7-29 Add new Section 7.9.2, "Flash Invalidation of L2 Cache" as follows:

The L2 cache may be completely invalidated by setting the L2I bit of the L2 control register (L2CTL). Note that no data is lost in this process because the L2 cache is a write-through cache and contains no modified data. Flash invalidation of the cache is necessary when the cache is initially enabled and may be necessary to recover from some error conditions such as a tag parity error.

The invalidation process requires several cycles to complete. The L2I bit remains set during this procedure and is then cleared automatically when the procedure is complete. The L2 cache controller issues retries for all transactions on the e500 core complex bus (CCB) while the flash invalidation process is in progress.

Note that the contents of memory-mapped SRAM regions of the data array are unaffected by a flash invalidation of the L2 cache regions of the array."

7.10, 7-33 Add new Section 7.10, "Initialization/Application Information," with the following sections: Section 7.10.1, "Initialization," with new subsections Section 7.9.1.1, "L2 Cache Initialization," and Section 7.9.1.2, "Memory-Mapped SRAM Initialization."

7.9.1.1 L2 Cache Initialization

After power-on reset, the valid bits in the L2 cache status array are in random states. Therefore, it is necessary to perform a flash invalidate operation before using the array as an L2 cache. This is done by writing a 1 to the L2I bit of the L2 control register (L2CTL). This can be done before or simultaneously with the write that enables the L2 cache. That is, the L2E and L2I bits of L2CTL can be set simultaneously. The L2I bit clears automatically, so no further writes are necessary.

7.9.1.2 Memory-Mapped SRAM Initialization

After power-on reset, the contents of the data and ECC arrays and are random, so all SRAM data must be initialized before it is read. If the cache is initialized by the core or any other device that uses sub-cacheline transactions, ECC error checking should be disabled during the initialization process to avoid false ECC errors generated during the read-modify-write process used for sub-cacheline writes to the SRAM array. This is done by setting the multi- and single-bit ECC error disable bits of the L2 error disable register (L2ERRDIS[MBECCDIS, SBECCDIS]). See Section 7.3.1.9.2, “Error Control and Capture Registers.” If the array is initialized by a DMA engine using cache-line writes, then ECC checking can remain enabled during the initialization process.

Also, add new Section 7.10.2, “Managing Errors,” with subsections Section 7.10.2.1, “ECC Errors,” and Section 7.10.2.2, “Tag Parity Errors.”

7.10.2.1 ECC Errors

An individual soft error that causes a single- or multi-bit ECC error can be cleared from the L2 array simply by executing a **dcbf** instruction for the address captured in the L2ERRADDR register. This will invalidate the line in the L2 cache. When the load that caused the ECC error is performed again, the data will be re-allocated into the L2 with ECC bits set properly again.

If the threshold for single bit errors set in the L2ERRCTL register is exceeded, then the L2 cache should be flash invalidated to clear out all single-bit errors.

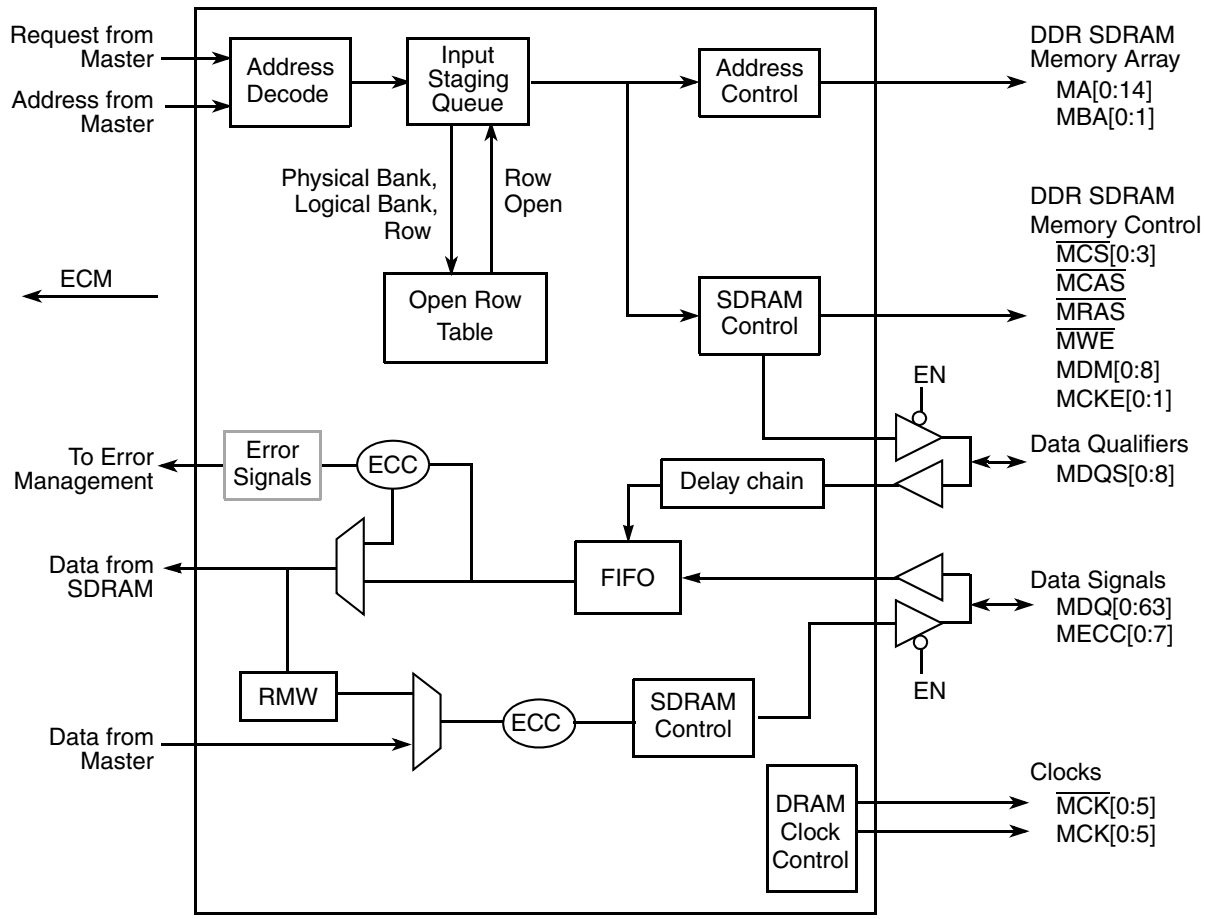
Note that no data is lost by executing **dcbf** instructions or flash invalidate operations because the L2 cache is write-through and contains no modified data.

7.10.2.2 Tag Parity Errors

A tag parity error must be fixed by flash invalidating the L2 cache. Note that executing a **dcbf** instruction for the address that caused the error to be reported is not sufficient because a tag parity error is seen as an L2 miss and does not cause invalidation of the bad tag. Proper L2 operation cannot be guaranteed if an L2 tag parity error is not repaired by a flash invalidation of the entire array.

9.1, 9-1 Remove the last sentence of the first paragraph.

9.1, 9-2 Replace Figure 9-1, “DDR Memory Controller Simplified Block Diagram,” with the following:



9.3.1, 9-4 In Table 9-1, remove signals MSYNC_IN and MSYNC_OUT.

9.3.2.2, 9-8 In Table 9-4, remove signals MSYNC_IN and MSYNC_OUT.

9.3.2.2, 9-8 In Table 9-4, correct the “MCKE” signal name to read “MCKE[0:1]”.

Also, change the first full sentence of the MCKE[0:1] signal description to read:

“Two identical output signals (each hereafter referred to simply as MCKE) used as the clock enable to one or more SDRAMs.”

9.3.2.2, 9-8 Correct timing description of MCK signals to read as follows:

“Source synchronous configuration as defined by the DDR_SDRAM_CLK_CNTL register determines timing relationship.”

9.4.1.4, 9-13 In Table 9-9, replace all instances of |CASLAT| with ⌈CASLAT⌉ .

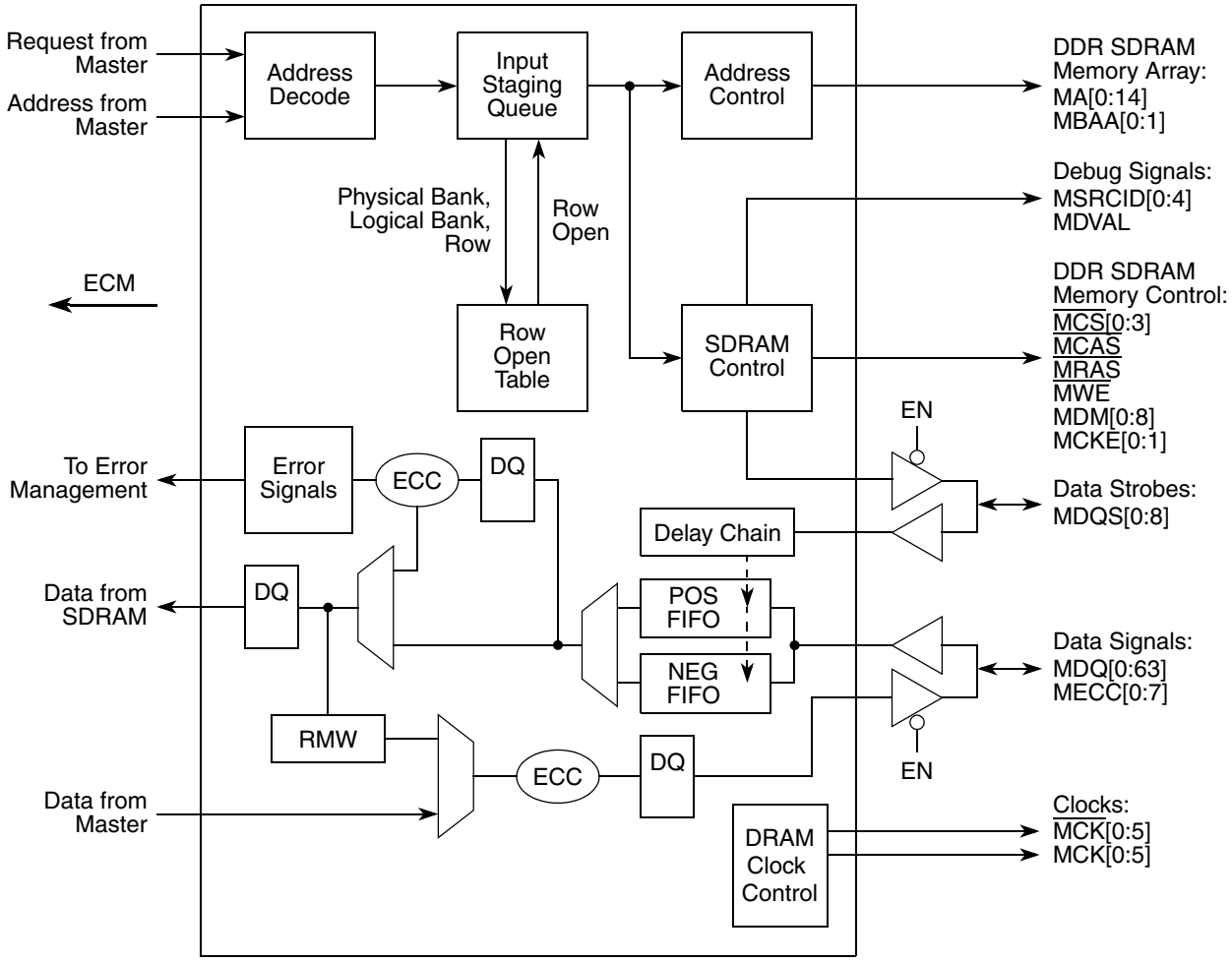
9.4.1.8, 9-14	Correct the DDR_SDRAM_CLK_CNTL[SS_EN] field description to read as follows: "Source synchronous enable. This bitfield must be set during initialization. See Section 9.6.1, "DDR SDRAM Initialization Sequence," for details. 0 Reserved 1 The address and command are sent to the DDR SDRAMs source synchronously."
9.4.1.5, 9-15	In Table 9-10, change the setting of the MBEE bit in the DDR_SDRAM_CFG[ECC_EN] description to read as follows: "ECC enable. Note that uncorrectable read errors may cause the assertion of core_fault_in, which causes the core to generate a machine check interrupt unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure an interrupt is generated. See Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled."
9.4.1.7, 9-16	In Table 9-12, add the following last sentence to the DDR_SDRAM_INTERVAL[REFINT] field description: "Note that REFINT must be set to a non-zero value in order for the DDR to enter sleep mode. See Section 17.5.1.5.3, "Sleep Mode," for additional details."
9.4.1.16, 9-22	Change the setting of MBEE bit in the ERR_DISABLE[MBED] description to read as follows: Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. Note that uncorrectable read errors cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBED must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
9.4.1.17, 9-23	Change the setting of the MBEE bit in the ERR_INT_EN[MBEE] description to read as follows: "Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and MBEE and

DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

0 Multiple-bit ECC errors cannot generate interrupts.

1 Multiple-bit ECC errors generate interrupts.”

9.5, 9-26 Replaced Figure 9-22, “DDR Memory Controller Block Diagram,” with the following figure:



9.5, 9-27 Remove Figure 9-23, “Controller DLL timing loop.”

9.5.1.1, 9-31 Delete the second-to-last paragraph, whose first sentence starts with the words, “If a disabled bank...”.

9.5.4, 9-36 In the last paragraph, change the phrase, “see Figure 9-27 for a single-beat read operation,” to read “see Figure 9-27 for a back-to-back burst read operation,”.

9.5.4–9.5.8, 9-36–9-40

In all burst examples, re-number the data beats in the figures to be D0, D1, D2, D3, D0, D1, D2, D3.

9.5.4.1, 9-37 Remove the second bulleted item.

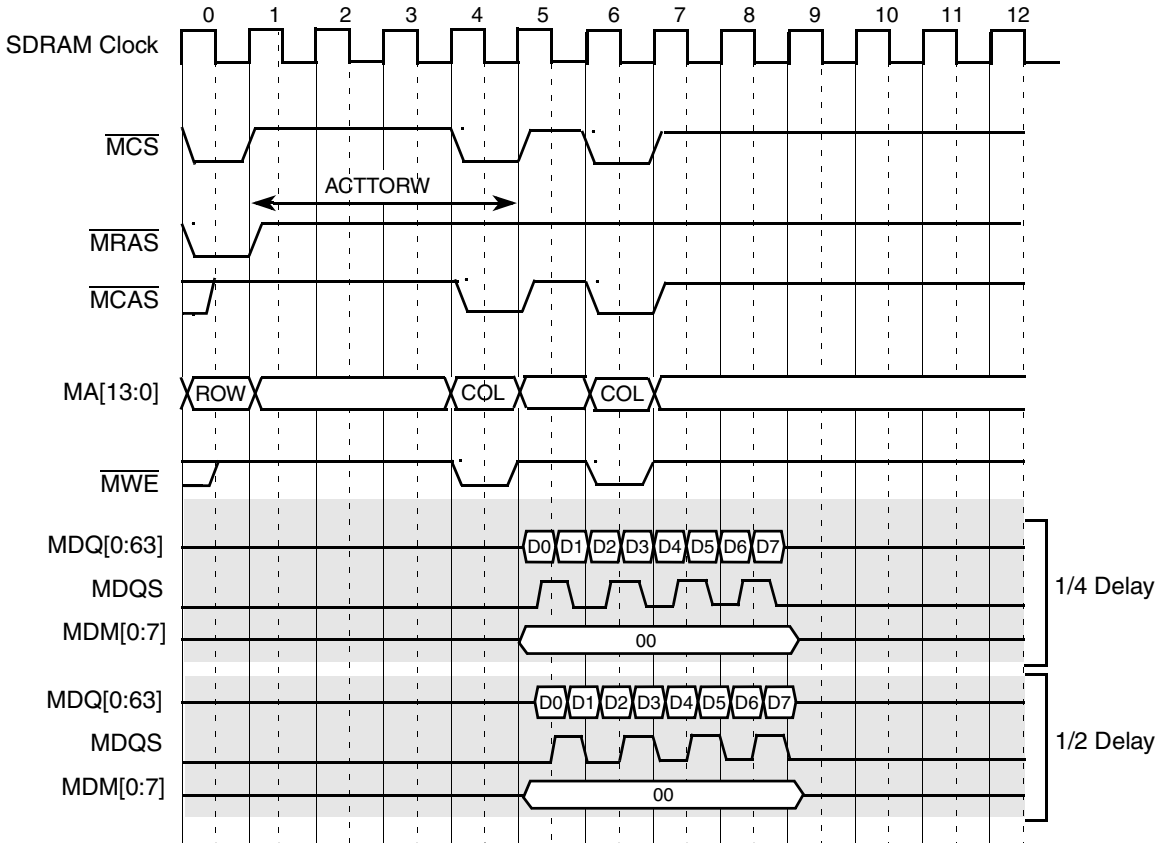
9.5.4.1, 9-37 Replace the third bulleted item (formerly the fourth; see above item) with the following:

"PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading."

9.5.4, 9-37 Change the title of Figure 9-29 to "DDR SDRAM Burst Write Timing—ACTTORW = 4".

9.5.6, 9-39 Change the last sentence of this section to read "Figure 9-32 shows the registered DDR SDRAM DIMM back-to-back burst write timing."

9.5.8, 9-40 Replace the existing Figure 9-33 with the figure below:



9.5.9, 9-41 In the paragraph preceding Section 9.5.9.1, change the reference to "Two sets of auto refresh commands..." to read "Three sets of auto refresh commands...". Also, change the reference to "...commands are also staggered in two groups..." to read "... commands are also staggered in three groups..."

9.5.9.1, 9-42 In Figure 9-34, remove the annotation of "0 or 3" in clock 12.

9.5.9.2, 9-42 After the second paragraph, add the following paragraph:

"All open pages are precharged before self refresh mode is entered."

	Also, correct the last sentence of the fifth paragraph to read: "This mode is controlled with DDR_SDRAM_CFG[DYN_PWR]."
9.5.10, 9-44	Replace the third and fourth sentences of this section with "If ECC is enabled for a sub-doubleword write transaction, a full read-modify-write is performed to properly update ECC bits. If ECC is disabled then no read-modify-write is required for sub-doubleword writes, and the data masks (MDM[0:8]) are used to prevent writing unwanted data to SDRAM."
9.5.12, 9-46	Add the following sentences before the second-to-last sentence of the second paragraph: "This read-modify-write operation is performed as an atomic transaction in the DDR controller. The write command is then issued 3-5 memory clocks after the completion of the read, depending on various system parameters."
9.5.13, 9-48	In the last paragraph, delete the three sentences that start and end with, "For all memory select errors..." and "...to show the transaction is not real."
9.6.1, 9-49	Replace the first paragraph with the following: "After configuration of all parameters is complete, system software must set DDR_SDRAM_CLK_CNTL[SS_EN] and DDR_SDRAM_CFG[MEM_EN] to properly enable the memory interface. Note that 200 μs must elapse after the memory clocks are stable (that is, initialization is complete of all clock related configuration registers) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. After MEM_EN has been set, the DDR memory controller automatically performs JEDEC-compliant initialization sequence to initialize memories according to the information in the SDMOCE and ESDMODE fields of the DDR_SDRAM_MODE register. The initialization sequence is as follows:" After the numbered list, add the following sentence: "Note that the BA0 and BA1 bits are automatically driven appropriately during the MODE REGISTER SET commands." Then the final sentence should be changed to, "After this automatic initialization is complete the memory array is ready for access and the memory controller begins processing memory transactions as they arrive."
10.2.2, 10-7	In Table 10-5, delete the last sentence in the Timing/Negation section for IRQ[0:11] signals about edge-sensitive interrupts.
10.3.7, 10-36	In Table 10-35, correct the who am I register name from "WHOAMI0" to "WHOAMI."
10.4.1, 10-41	Add the following new paragraph after the first paragraph: "Note that the IPR, IS, and IRR are internal registers that are not accessible to the programmer."

11.3, 11-4 — In Table 11-3, correct the I2CDFSRR reset value to be 0x10.

11.4.5.2, 11-18 — Insert the following sentence after the polynomial used for CRC calculation in the fourth paragraph:

"CRC values are calculated using the above polynomial with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000."

12.3.1.9, 12-15 — In Table 12-16, change the second sentence of the asserted ("1") description for the ULSR[BI] to read:

"A break condition is expected to last at least two character lengths and a new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected."

13.1.3.1, 13-3 — Change the third sentence of the first paragraph to read:

"In addition to establishing the frequency of the external local bus clock, CLKDIV also affects the resolution of signal timing shifts in GPCM mode, and the interpretation of UPM array words in UPM mode."

13.1.3.1, 13-4 — In the last sentence, change "LCLK[0:3]" to "LCLK[0:2]."

13.2, 13-6, 13-7, and 13-9 — In Table 13, change all instances of "RAS" to "$\overline{\text{RAS}}$" and all instances of CAS to $\overline{\text{CAS}}$.

13.3.1.2.3, 13-16 and 13-17 — In Table 13-7, Table 13-8, Figure 13-4, and Figure 13-5, change bits 17–18 to reserved.

13.4.3.7.3, 13-55 — In the section title, change "CAS" to "$\overline{\text{CAS}}$".

13.4.4, 13-61 — Insert the following footnote associated with the word 'signals' in the second sentence of the section:

"If the LGPL4/LGTA/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pullup is required. Refer to the hardware specification for details regarding termination options."

13.4.4.2, 13-65 — Add the following third paragraph:

"Note that the UPM memory region must be cache-inhibited or write-through (the MMU page must have the I or W bit set) during the time that the UPM array is being written. If the memory is to be cacheable and/or copy-back, the MMU must be set accordingly after the UPM array is initialized."

14.5.2, 14-14 — In Table 14-3, add the following as the first row under the TSEC1 FIFO Control and Status Registers section:

0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	14.5.3.2.1/14-30

14.5.2, 14-15 — In Table 14-3, correct the reset value for the OSTBD register to 0x0800_0000.

14.5.2, 14-16	In Table 14-3, correct the MACCFG1 register from having R/W, R access to R/W access.
14.5.2, 14-16	In Table 14-3, correct the IFSTAT register from having R access to R/W access.
14.5.2, 14-18	In Table 14-3, correct the CAR1 and CAR2 registers from having R access to R/W access.
14.5.2, 14-19	In Table 14-3, correct the ATTR and ATTRELI registers from having R access to R/W access.
14.5.3.1.1, 14-21	In Table 14-4, correct the IEVENT[RXC] bitfield description to read as follows: “Receive control interrupt. A control frame was received. If MACCFG1[Rx_Flow] is set, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was received. 0 Control frame not received 1 Control frame received”
14.5.3.1.3, 14-24 and14-25	Undocument bit fields TXEDIS, CRL/XDADIS, and XFUNDIS.
14.5.3.1.7, 14-28	Remove bit field DMACTRL[TOD] from Figure 14-12 and Table 14-10. This bit position (29) is now reserved.
14.5.3.2.1, 14-30	Insert the following section as the first subsection of Section 14.5.3.2:

14.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 14-14, is writable by the user to configure the properties of the TSEC FIFO.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16													29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TFC_PAUSE_EN	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_404C; TSEC2:0x2_504C															

Figure 14-14. FIFO_PAUSE_CTRL Register Definition

Table 14-12 describes the fields of the FIFO_PAUSE_CTRL register.

Table 14-12. FIFO_PAUSE_CTRL Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	TFC_PAUSE_EN	TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset. 0 Pause control frame transmission disabled. 1 Pause control frame transmission enabled.
31	—	Reserved

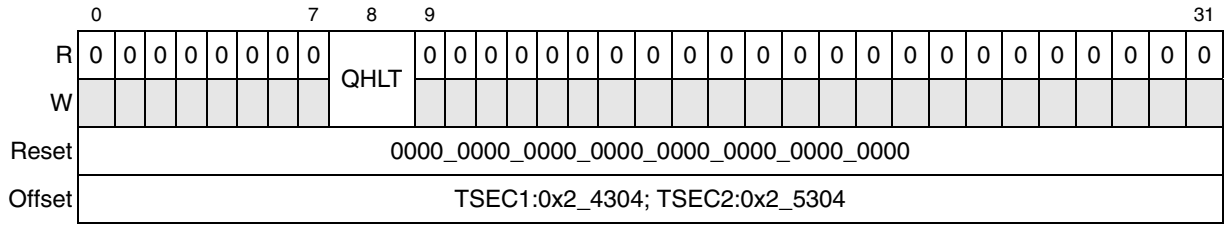
14.5.3.3.1, 14-33 In Table 14-15, replace the bitfield description of TCTRL[TFC_PAUSE] with the following:

"Transmit flow control pause frame. Use this bit to transmit a PAUSE frame. To transmit a flow control pause frame, first set FIFO_PAUSE_CTRL[TFC_PAUSE_EN]. Next, set MACCFG1[GTS]. If TFC_PAUSE is then set, the MAC stops transmission of data frames after the current transmission completes. The GTSC interrupt in the IEVENT register is asserted. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame with the duration value obtained from the PTV register. The TXC interrupt occurs after sending the control pause frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. Note that if the transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, the MAC may still transmit a MAC control PAUSE frame.

0 No outstanding pause frame transmission request.

1 Pause frame transmission requested."

14.5.3.4.2, 14-41 Replace Figure 14-27 with the following figure:



14.5.3.6.1, 14-49 In Table 14-32, add the following sentence to the MACCFG1[Rx_EN] field description:

"If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set)."

14.5.3.6.1, 14-49 In Table 14-32, add the following sentence to the MACCFG1[Tx_EN] field description:

"If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful transmit stop interrupt (IEVENT[GTSC] is set)."

14.5.3.6.6, 14-53 In Table 14-37, clarify the bit description of the Mgmt Clock Select field of the MIIMCFG register so that instead of concluding with "... divided by eight." the third sentence concludes with "... divided first by eight and then further divided by the following value:".

14.5.3.6.8, 14-55 In Table 14-39, replace "(0 is reserved)" with the following:

"At least one PHY address is reserved for the TBI PHY address as programmed in the TBIPA register (Section 14.5.3.1.8, "TBI Physical Address Register (TBIPA)," Figure 14-13). The default of the TBI PHY address is 0x00. The user must be sure to assign a physical address to the TBI so as to not conflict with the external PHY physical address as discussed in the register initialization steps in Section 14.7, "Initialization/Application Information."

14.5.3.6.12, 14-56 and 14-56

Delete the first sentence and replace Figure 14-45 with the following figure (exposing the Link Fail status field):

	0																					21	22	23					28		31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Excess Defer	0	0	0	0	0	Link Fail	0	0	0
W																																
Reset	0000_0000_0000_0000_0000_0000_0000_0000																															
Offset	TSEC1:0x2_453C; TSEC2:0x2_553C																															

14.5.3.7.44, 14-80 Correct the first sentence to read:
"Carry register bits are cleared when written with a one."

14.5.3.7.45, 14-82 Correct the first sentence to read:
"Carry register bits are cleared when written with a one."

14.5.3.7.47, 14-83 In Figure 14-94, change bit 30 to be "Reserved".

14.6.2.2, 14-110 Insert the following step between steps 3 and (formerly) 4:
"4. Wait for a period of 9.6 Kbytes worth of data on the interface (~8 ms worst case)."

14.6.2.2, 14-111 Correct step 11 of the graceful reset procedure to read:
"11. Set WWR and WOP bits in DMACTRL register"

14.6.2.3, 14-111 Replace the second paragraph with the following:
"If the user has a frame ready to transmit, a transmit-on-demand function may be emulated while in polling mode by using the graceful-transmit-stop feature. First, clear the IMASK[GTSCEN] bit to mask the graceful-transmit-stop complete

interrupt. Next set, then immediately clear the DMACTRL[GTS] bit. Clear the resulting IEVENT[GTSC] bit. Finally, the IMASK[GTSCEN] bit may be set once again."

14.6.2.3, 14-112 Insert the following sentence between the third and (formerly) fourth sentences of the sixth paragraph:
"The pause duration is defined by the received pause control frame and begins when the frame was first received."

14.6.2.6.4, 14-116 Add the following section after Section 14.6.2.6.3:

14.6.2.6.3 CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value for the purposes of TSEC. The RFC 3309 algorithm uses the following polynomial to calculate the CRC value:
x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x1+x0 or 0x04c11db7.

Given a destination MAC address of DA=01000CCCCCCC, the algorithm results in a CRC remainder value of 0xA29F4BBC.

Bit-reversing the low-order byte of the CRC value (0xBC) yields BR_CRC = 0x3D = 0b00111101

The high-order 3-bits of the new BR_CRC value are used to select which 32-bit register (of the 8) to use. This example maps the DA to register 1.

High-order 3 bits of BR_CRC: HO_CRC = 0b001 = 1

The low-order 5 bits are used to select which bit to set in the given register (with a value of 0 setting 0x8000_0000 and 31 setting 0x0000_0001). Therefore, the example DA maps to bit 29 of register 1.

Low-order 5 bits of BR_CRC: LO_CRC = 0b11101 = 29

Therefore, GADDR1 is ORed with the value 0x0000_0004.

Additional calculated examples follow:

Example 1:

- Destination MAC address: DA = 01005E000128
- CRC remainder value: CRC = 0x821D6CD3
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xCB = 0b11001011
- High-order 3 bits of BR_CRC: HO_CRC = 0b110 = 6
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01011 = 11
- GADDR6 = 0x0010_0000

Example 2:

- Destination MAC address: DA = 0004F0604F10
- CRC remainder value: CRC = 0x1F5A66B5
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xAD = 0b10101101
- High-order 3 bits of BR_CRC: HO_CRC = 0b101 = 5
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01101 = 13
- GADDR5 = 0x0004_0000

14.6.2.8, 14-118 In Table 14-115, correct the RXC description to read as follows:

"Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was first received." 14.6.3, 14-122Between the second and third paragraph, add the following:
"The status field of the BD is 16-bit field, as is the length field. The data buffer pointer is a 32-bit field. Therefore, the BDs should be accessed with the following C structure:

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct bd_struct {
 uint_16 flags;
 uint_16 length;
 uint_32 bufptr;
};"
```

14.7.1.1, 14-131 In Table 14-123, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

14.7.1.2, 14-134 In Table 14-126, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

14.7.1.3, 14-138 In Table 14-129, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

14.7.1.4, 14-142 — In Table 14-132, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

14.7.1.5, 14-146 — In Table 14-135, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

15.2.2, 15-6 and 15.4.1.3, 15-29 — In the third bullet of the list of signals defined for the external control interface, clarify that the falling edge of DMA_DREQ sets MRn[CS] and for DMA_DDONE with the following:
DMA_DDONE assertion indicates that the DMA engine has completed the transfer. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

15.3.1, 15-6 — In Table 15-4, add the following memory locations and designate them as 'Reserved':

0x2_112C

0x2_1148–0x2_117C

0x2_11AC

0x2_11C8–0x2_11FC

0x2_122C

0x2_1248–0x2_127C

0x2_12AC

0x2_12C8–0x2_12FC

15.3.2.10, 15-20 — In Figure 15-15, correct the offset value for CLSDAR2 to 0x2_1234.

15.3.2.11, 15-20 — In Figure 15-16, correct the offset value for the NLSDAR*n*–DMA 2 next list descriptor address register to 0x2_123C.

15.4.1.1.1, 15-25 — Change step 1 in the sequence to read, "Poll the channel state (see Table 15-19) to confirm that the specific DMA channel is idle."

15.4.1.1.2, 15-25 — Change step 1 in the sequence to read, "Poll the channel state (see Table 15-19) to confirm that the specific DMA channel is idle."

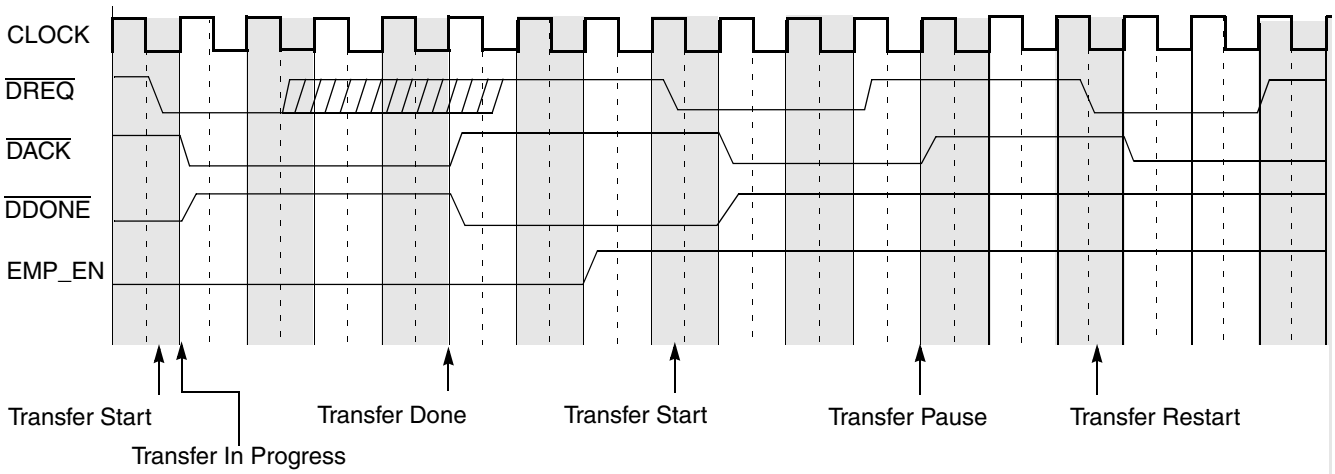
15.4.1.1.3, 15-26 — Change step 2 in the sequence to read, "Poll the channel state (see Table 15-19) to confirm that the specific DMA channel is idle."

15.4.1.1.4, 15-27 — Change step 3 in the sequence to read, "Poll the channel state (see Table 15-19) to confirm that the specific DMA channel is idle."

15.4.1.2.3, 15-28 Change step 2 in the sequence to read, “Poll the channel state (see Table 15-19) to confirm that the specific DMA channel is idle.”

15.4.1.2.4, 15-28 Change step 3 in the sequence to read, “Poll the channel state (see Table 15-19) to confirm that the specific DMA channel is idle.”

15.4.1.3, 15-30 Replace Figure 15-20 with the following, showing more flexible negation of $\overline{\text{DREQ}}$:



16.3.1.2, 16-21 Add the following sentence to the end of the third paragraph:

“Note that outbound translation windows must not overlap the configuration access registers.”

16.3.1.2.4, 16-25 In Figure 16-9, add a footnote to EN field (bit 0) that states, “For POWAR0, translation is always enabled. The enable field (EN) may be read and written, but the value is ignored.”

16.3.1.4, 16-32 Insert the following between the fourth and fifth paragraphs:

“If a data parity error occurs during an inbound configuration write access, the error is reported and captured. However, the erroneous data is written to the register specified in the transaction. Therefore, PCI data parity error recovery routines must include reinitialization of the PCI configuration register if the error occurred during a configuration write.”

16.3.1.4.7, 16-38 Add the following sentence before Figure 16-21:

“Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.”

16.3.1.4.8, 16-38 Add the following sentence before Figure 16-21:

“Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.”

16.3.2.19, 16-53 In Table 16-45, change the description for bit 0 to read as follows:

"PCI agent/host. Read-only. Indicates the reset value of the PCI host/agent configuration signal, $\overline{\text{LWE2}}$.

0 PCI interface is in host mode

1 PCI interface is in agent mode"

16.4.2.11.2–16.4.2.11.3, 16-73–16-75

Replace the existing sections with the following:

16.4.2.11.2 Accessing the PCI Configuration Space in Host Mode

To access the configuration space, a 32-bit value must be written to the PCI CFG_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. Note that the Bus Master bit in the MPC8555E PCI bus command register must be set before an outbound configuration access is attempted. Device 0 on PCI bus 0 is the MPC8555E itself; thus, device 0, bus 0 is used to access the internal PCI configuration header.

When the MPC8555E detects an access to PCI CFG_DATA, it checks the enable flag and the device number in the PCI CFG_ADDR register. If the enable bit is set, and the device number is not 0b1_1111, the MPC8555E performs a configuration cycle translation function and runs a configuration-read or configuration-write transaction on the PCI bus. If the bus number corresponds to the local PCI bus (bus number = 0x00), the MPC8555E performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the MPC8555E performs a type 1 configuration cycle translation. The device number 0b1_1111 is used for performing interrupt-acknowledge and special-cycle transactions. See Section 16.4.2.12, "Other Bus Transactions," for more information.

See Section 16.3.1.1.1, "PCI Configuration Address Register (CFG_ADDR)," for details on PCI CFG_ADDR and Section 16.3.1.1.2, "PCI Configuration Data Register (CFG_DATA)," for details on PCI CFG_DATA.

Note that because all PCI registers are intrinsically little-endian, in the following examples, the data in the configuration register is shown in little-endian order. PowerPC processor accesses to the PCI CFG_DATA register should use the load/store with byte-reversed instructions. External PCI masters that use the local address map to access configuration space do not need to reverse bytes since byte lane redirection from the little-endian PCI bus is performed internally.

Example: Configuration sequence, 4-byte data read from the revision ID/standard programming interface/subclass code/class code registers at address offset 0x08 of the PCI configuration header (device 0 on the PCI bus 0 is the MPC8555E itself).

```
Initial values:
r0 contains 0x8000_0008
r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR register)
r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA register)
r3 contains 0xFFFF_FFFF
Register at 0x08 contains 0x0B20_0002 (0x0B to 0x08)

Code sequence:
stw r0, 0 (r1)
lwbrx r3, 0 (r2)

Results:
Address CCSRBAR + 0x0_8000 contains 0x8000_0008
Register r3 contains 0x0B20_0002
```

Example: Configuration sequence, 4-byte data write to PCI register at address offset 0x14 of Device 1 on PCI bus 0.

```
Initial values:
     r0 contains 0x8000_0814
     r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR register)
     r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA register)
     r3 contains 0x1122_3344
     Register at 0x14 contains 0xFFFF_FFFF (0x17 to 0x14)

Code sequence:
     stw r0, 0 (r1)// Update PCI CFG_ADDR register to point to
     //register offset 0x14 of device 1.
     stwbrx r3, 0 (r2)

Results:
     Address CCSRBAR + 0x0_8000 contains 0x8000_0814
     Register at 0x14 contains 0x1122_3344 (0x17 to 0x14)
```

Example: Configuration sequence, 2-byte data write to PCI register at address offset 0x1C of Device 1 on PCI bus 0.

```
Initial values:
     r0 contains 0x8000_081C
     r1 contains CCSRBAR + 0x0_8000
     r2 contains CCSRBAR + 0x0_8004
     r3 contains 0xDDCC_BBAA
     Register at 0x1C contains 0xFFFF_FFFF (0x1F to 0x1C)

Code sequence:
     stw r0, 0 (r1)
     sthbrx r3, 0 (r2)

Results:
     Address CCSRBAR + 0x0_8000 contains 0x8000_081C
     Register at 0x1C contains 0xFFFF_BBAA (0x1F to 0x1C)
```

16.4.2.11.3 PCI Configuration in Agent and Agent Lock Modes

In general, agents should not access the configuration space of other external PCI devices. Configuration of agents is a function usually reserved for the host. When the MPC8555E is in agent mode, it responds to remote host-generated PCI configuration cycles. This occurs when a configuration command is decoded along with the IDSEL input signal being asserted. When the MPC8555E is in

	agent lock mode, it retries all externally-generated PCI configuration cycles until the ACL bit in the PCI bus function register (0x44) is set. See Section 16.5.1, “Power-On Reset Configuration Modes,” for more information. In either agent or agent lock mode, access to the internal PCI configuration header by the processor core is handled as described in Section 16.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” using device = 0 and bus = 0 in PCI CFG_ADDR to indicate the internal PCI header.
17.3.4, 17-23 and 24	In Figure 17-6 and Table 17-8, change the bitfield “N” to be bit position 23 (formerly 24). Subsequently, describe bit positions 24 through 31 as “Reserved.”
18.4, 18-4	In Table 18-3, change the reset value of the GPINDR register to be 0x*nnnn*_0000.
18.4, 18-4	In Table 18-3, remove the second-to-last row formerly describing the DDR DLL control register (DDRDLLCR).
18.4.1.9, 18-12	In Figure 18-9, change the reset value of the GPINDR register to be *nnnn_nnnn_nnnn_nnnn*_0000_0000_0000_0000.
18.4.1.14, 18-19	In Table 18-17, change the bit ranges “32–47” to “0–15” and “48–63” to “16–31”.
18.4.1.17, 18-20	Remove Section 18.4.1.17, “DDR DLL Control Register (DDRDLLCR).”
18.5.1.5.3, 18-25	Add the following sentence to the end of the first paragraph: “Note that the DDR controller does not shut down unless DDR_SDRAM_INTERVAL[REFINT] is set to a non-zero value. See Section 9.4.1.7, “DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL),” for details.”
19.3.1, 19-4	In Table 19-1, correct the name of the register with offset 0xE_1094 from PMLCB5 to PMLCB8.
20.1, 20-2	In Figure 20-1, remove PCI interface to trace buffer.
20.3.1.2, 20-14	Change the last sentence in the first paragraph to, “Note that the transaction address is qualified with the bits described in Section 20.3.1.3, “Watchpoint Monitor Address Mask Register (WMAMR),” before being compared with WMAR. Note also that the contents of WMAR are not qualified with WMAMR.”
20.3.1.3, 20-15	Change the last part of the first sentence to be “...contains the mask that is applied to a transaction address before the address is compared with WMAR.”
20.3.2.2, 20-20	In the first sentence, replace “TBCR[AMD]” with “TBCR0[AMD]”. Also, replace the last sentence of the first paragraph with the following, “The transaction address is qualified with the bits described in Section 20.3.2.3, “Trace Buffer Address Mask Register (TBAMR),” before being compared with TBAR. Note that the contents of TBAR are not qualified with TBAMR.”
20.3.2.2, 20-20	Change the last part of the first sentence to be “...contains the mask that is applied to a transaction address before the address is compared with TBAR.”

20.3.2.2, 20-21	In the last sentence of the first paragraph, replace “TBCR[TMD]” with “TBCR0[TMD]”.
20.4.6.1, 20-31 and 20-32	Change all instances of “TRSEL” and “TRSEL1” to “IFSEL” and all instances of “TBCR” to “TBCR1”.
21.3.1, 21-26	In Table 21-9, delete “Refer to Table 13-7” in the SBC field description.
21.3.1.1, 21-27	In Table 21-10, replace footnote 1 with “See SBC for FCC1 and FCC2, Table 21-9”.
24.4.1, 24-6	In Table 24-2, change the name of bits 6–7 to read “MAD4–MAD3”. Also, replace the description with the following: Master address output pin x connection. Note that the address indexes are relative to FCC1; see Figure 24-5. These bits determine the number of ATM PHYs supported by FCC1 and FCC2. 00 FCC1 supports 7 PHYs; FCC2 supports 31 PHYs 01 FCC1 supports 15 PHYs; FCC2 supports 15 PHYs 10 Reserved 11 FCC1 supports 31 PHYs; FCC2 supports 7 PHYs
23.6.1, 23-17	In Figure 23-10, change the offset (Addr) for SI2GMR from “0x9_1B28” to “0x9_1B48”.
24.4.1, 24-7	In Figure 24-5, correct the pin name for FCC1 UTOPIA Rx address RxAddr[3] = PD9 to RxAddr[3] = PD29.
24.4.1, 24-7	In Figure 24-5, correct the pin name for FCC2 UTOPIA Rx address RxAddr[1] = PC23 to RxAddr[1] = PD23.
26.1, 26-2	Replace the first three bullets with the following: • The maximum input clock is the internal CPM clock which is the core complex bus (CCB) clock divided by 3. • Maximum period of 2.6 seconds (at 100 MHz) • 10-ns resolution (at 100 MHz)
26.2, 26-2	Replace the first two bullets with: • Internal CPM clock (CCB/3) • Internal CPM clock divided by 16 (CCB/48) Also, replace the first four sentences of the following paragraph with: “The internal CPM clock is generated in the CPM clock synthesizer and defaults to the CCB clock frequency divided by 3. The user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer.”

Also, replace the last two sentences of the next paragraph with, “The best resolution of the timer is one clock cycle (10 ns at 100 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (2.6 seconds at 100 MHz).”

26.2, 26-3 — Replace the Note toward the bottom of the page with, “$\overline{\text{TGATE}}x$ is internally synchronized to the internal CPM clock. After the falling edge of $\overline{\text{TGATE}}x$ is recognized, the counter begins counting after one internal CPM clock cycle when working with the internal clock.”

41.11, 41-82 — Add the following clarifying note after the second paragraph:

NOTE

Ensure that the transmit INTQs and the receive INTQs are programmed as separate queues.

43.5, 43-12 — In Table 43-5, replace IMMR with CCSRBAR in footnote 1.

44.5, 44-12 — In Table 44-6, replace IMMR with CCSRBAR in footnote 1.

Glossary

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this reference manual.

A

Architecture. A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Atomic access. A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The Power Architecture technology implements atomic accesses through the **lwarx**/**stwcx.** instruction pair.

Autobaud. The process of determining a serial data rate by timing the width of a single bit.

B

Beat. A single state on the bus interface that may extend across multiple bus cycles. A transaction can be composed of multiple address or data *beats*.

Big-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *most-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the *most-significant byte*. See *Little endian*.

Boundedly undefined. A characteristic of certain operation results that are not rigidly prescribed by the Power Architecture technology. Boundedly-undefined results for a given operation may vary among implementations and between execution attempts in the same implementation.

Although the architecture does not prescribe the exact behavior for when results are allowed to be boundedly undefined, the results of executing instructions in contexts where results are allowed to be boundedly undefined are constrained to ones that could have been achieved by executing an arbitrary sequence of defined instructions, in valid form, starting in the state the machine was in before attempting to execute the given instruction.

Breakpoint. A programmable event that forces the core to take a breakpoint exception.

Burst. A multiple-beat data transfer whose total size is typically equal to a cache block.

Bus clock. Clock that causes the bus state transitions.

Bus master. The owner of the address or data bus; the device that initiates or requests the transaction.

C

Cache. High-speed memory containing recently accessed data or instructions (subset of main memory).

Cache block. A small region of contiguous memory that is copied from memory into a *cache*. The size of a cache block may vary among processors; the maximum block size is one *page*. In Power Architecture processors, *cache coherency* is maintained on a cache-block basis. Note that the term 'cache block' is often used interchangeably with 'cache line.'

Cache coherency. An attribute wherein an accurate and common view of memory is provided to all devices that share the same memory system. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor's cache.

Cache flush. An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to main memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

Caching-inhibited. A memory update policy in which the *cache* is bypassed and the load or store is performed to or from main memory.

Cast out. A *cache block* that must be written to memory when a cache miss causes a cache block to be replaced.

Changed bit. One of two *page history bits* found in each *page table entry* (PTE). The processor sets the changed bit if any store is performed into the *page*. See also *Page access history bits* and *Referenced bit*.

Clean. An operation that causes a cache block to be written to memory, if modified, and then left in a valid, unmodified state in the cache.

Clear. To cause a bit or bit field to register a value of zero. See also *Set*.

Context synchronization. An operation that ensures that all instructions in execution complete past the point where they can produce an *exception*, that all instructions in execution complete in the context in which they began execution, and that all subsequent instructions are *fetched* and executed in the new context. Context synchronization may result from executing specific instructions (such as **isync** or **rfi**) or when certain events occur (such as an exception).

Copy-back operation. A cache operation in which a cache line is copied back to memory to enforce cache coherency. Copy-back operations consist of snoop push-out operations and cache cast-out operations.

D

Direct-mapped cache. A cache in which each main memory address can appear in only one location within the cache; operates more quickly when the memory request is a cache hit.

Double data rate. Memory that allows data transfers at the start and end of a clock cycle. thereby doubling the data rate.

E

Effective address (EA). The 32-bit address specified for a load, store, or an instruction fetch. This address is then submitted to the MMU for translation to either a *physical memory* address or an I/O address.

Exclusive state. MEI state (E) in which only one caching device contains data that is also in system memory.

F

Fetch. Retrieving instructions from either the cache or main memory and placing them into the instruction queue.

Flush. An operation that causes a cache block to be invalidated and the data, if modified, to be written to memory.

Frame-check sequence (FCS). Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD, and CRC.

G

General-purpose register (GPR). Any of the 32 registers in the general-purpose register file. These registers provide the source operands and destination results for all integer data manipulation instructions. Integer load instructions move data from memory to GPRs and store instructions move data from GPRs to memory.

Gigabit media-independent interface (GMII) sublayer. Sublayer that provides a standard interface between the MAC layer and the physical layer for 1000-Mbps operation. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.

Guarded. The guarded attribute pertains to out-of-order execution. When a page is designated as guarded, instructions and data cannot be accessed out-of-order.

H

Harvard architecture. An architectural model featuring separate caches and other memory management resources for instructions and data.

I

Illegal instructions. A class of instructions that are not implemented for a particular processor. These include instructions not defined by the architecture. In addition, for 32-bit implementations, instructions that are defined only for 64-bit implementations are considered to be illegal instructions. For 64-bit implementations instructions that are defined only for 32-bit implementations are considered to be illegal instructions.

Implementation. A particular processor that conforms to the architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features.

Inbound ATMU windows. Mappings that perform address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and map the transaction to its target interface.

In-order. An aspect of an operation that adheres to a sequential model. An operation is said to be performed in-order if, at the time that it is performed, it is known to be required by the sequential execution model.

Integer unit. An execution unit in the core responsible for executing integer instructions.

Inter-packet gap. The gap between the end of one Ethernet packet and the beginning of the next transmitted packet.

Instruction latency. The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

K

Kill. An operation that causes a *cache block* to be invalidated without writing any modified data to memory.

L

L2 cache. Level-2 cache. See *Secondary cache*.

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that execution for a subsequent instruction.

Least-significant bit (lsb). The bit of least value in an address, register, field, data element, or instruction encoding.

Least-significant byte (LSB). The byte of least value in an address, register, data element, or instruction encoding.

Little-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *least-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the *most-significant byte*. See *Big endian*.

Local access window. Mapping used to translate a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. The local memory map is defined by a set of eight local access windows. The size of each window can be configured from 4 Kbytes to 2 Gbytes.

M

Media access control (MAC) sublayer. Sublayer that provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.

Media-independent interface (MII) sublayer. Sublayer that provides a standard interface between the MAC layer and the physical layer for 10/100-Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.

Medium-dependent interface (MDI) sublayer. Sublayer that defines different connector types for different physical media and PMD devices.

Memory access ordering. The specific order in which the processor performs load and store memory accesses and the order in which those accesses complete.

Memory-mapped accesses. Accesses whose addresses use the page or block address translation mechanisms provided by the MMU and that occur externally with the bus protocol defined for memory.

Memory coherency. An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.

Memory consistency. Refers to agreement of levels of memory with respect to a single processor and system memory (for example, on-chip cache, secondary cache, and system memory).

Memory management unit (MMU). The functional unit that is capable of translating an *effective* (logical) *address* to a physical address, providing protection mechanisms, and defining caching methods.

Modified/exclusive/invalid (MEI). *Cache coherency* protocol used to manage caches on different devices that share a memory system. Note that the PowerPC architecture does not specify the implementation of a MEI protocol to ensure cache coherency.

Modified state. MEI state (M) in which one, and only one, caching device has the valid data for that address. The data at this address in external memory is not valid.

Most-significant bit (msb). The highest-order bit in an address, registers, data element, or instruction encoding.

Most-significant byte (MSB). The highest-order byte in an address, registers, data element, or instruction encoding.

N

NaN. An abbreviation for not a number; a symbolic entity encoded in floating-point format. There are two types of NaNs—signaling NaNs and quiet NaNs.

No-op. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

O

OCeaN. (On-chip network) Non-blocking crossbar switch fabric. Enables full duplex port connections at 128Gb/s concurrent throughput and independent per port transaction queuing and flow control. Permits high bandwidth, high performance, as well as the execution of multiple data transactions.

Outbound ATMU windows. Mappings that perform address translations from local 32-bit address space to the address spaces of PCI, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.

P

Packet. A unit of binary data that can be routed through a network. Sometimes packet is used to refer to the frame plus the preamble and start frame delimiter (SFD).

Page. A region in memory. The OEA defines a page as a 4-Kbyte area of memory aligned on a 4-Kbyte boundary.

Page access history bits. The *changed* and *referenced* bits in the PTE keep track of the access history within the page. The referenced bit is set by the MMU whenever the page is accessed for a read or write operation. The changed bit is set when the page is stored into. See *Changed bit* and *Referenced bit*.

Page fault. A page fault is a condition that occurs when the processor attempts to access a memory location that does not reside within a *page* not currently resident in *physical memory*. A page fault exception condition occurs when a matching, valid *page table entry* (PTE[V] = 1) cannot be located.

Page table. A table in memory is comprised of *page table entries*, or PTEs. It is further organized into eight PTEs per PTEG (page table entry group). The number of PTEGs in the page table depends on the size of the page table (as specified in the SDR1 register).

Page table entry (PTE). Data structures containing information used to translate *effective address* to physical address on a 4-Kbyte page basis. A PTE consists of 8 bytes of information in a 32-bit processor and 16 bytes of information in a 64-bit processor.

Physical coding sublayer (PCS). Sublayer responsible for encoding and decoding data stream to and from the MAC sublayer. Medium (1000BASEX) 8B/10B coding is used for fiber. Medium (1000BASET) 8B1Q coding is used for unshielded twisted pair (UTP).

Physical medium attachment (PMA) sublayer. Sublayer responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices (SERDES) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers. For fiber medium (1000BASEX) the interface on the PMD side of the PMA is a one-bit 1250 MHz signal, while on the PMA's PCS side the interface is a ten-bit interface (TBI) at 125 MHz. The TBI is an alternative to the GMII interface. If the TBI is used the gigabit Ethernet controller must be capable of performing the PCS function. For UTP medium, the PMD interface side of the PMA consists of four pair of 62.5-MHz PAM5 encoded signals, while the PCS side provides the 1250-Mbps input to a 8B1Q4 PCS.

Physical medium dependent (PMD) sublayer. Sublayer responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.

Physical memory. The actual memory that can be accessed through the system's memory bus.

Pipelining. A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

Primary opcode. The most-significant 6 bits (bits 0–5) of the instruction encoding that identifies the type of instruction.

Program order. The order of instructions in an executing program. More specifically, this term is used to refer to the original order in which program instructions are fetched into the instruction queue from the cache.

Protection boundary. A boundary between *protection domains*.

Protection domain. A protection domain is a segment, a virtual page, a BAT area, or a range of unmapped effective addresses. It is defined only when the appropriate relocate bit in the MSR (IR or DR) is 1.

Q

Quad word. A group of 16 contiguous locations starting at an address divisible by 16.

Quiesce. To come to rest. The processor is said to quiesce when an exception is taken or a **sync** instruction is executed. The instruction stream is stopped at the decode stage and executing instructions are allowed to complete to create a controlled context for instructions that may be affected by out-of-order, parallel execution. See *Context synchronization*.

R

rA. The **r**A instruction field is used to specify a GPR to be used as a source or destination.

rB. The **r**B instruction field is used to specify a GPR to be used as a source.

rD. The **r**D instruction field is used to specify a GPR to be used as a destination.

rS. The **r**S instruction field is used to specify a GPR to be used as a source.

Record bit. Bit 31 (or the Rc bit) in the instruction encoding. When it is set, updates the condition register (CR) to reflect the result of the operation.

Reconciliation sublayer. Sublayer that maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.

Reduced instruction set computing (RISC). An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

Referenced bit. One of two *page history bits* found in each *page table entry*. The processor sets the *referenced bit* whenever the page is accessed for a read or write. See also *Page access history bits*.

Reservation. The processor establishes a reservation on a *cache block* of memory space when it executes an **lwarx** instruction to read a memory semaphore into a GPR.

Reservation station. A buffer between the dispatch and execute stages that allows instructions to be dispatched even though the results of instructions on which the dispatched instruction may depend are not available.

S

Secondary cache. A cache memory that is typically larger and has a longer access time than the primary cache. A secondary cache may be shared by multiple devices. Also referred to as L2, or level-2, cache.

Set (*v*)**.** To write a nonzero value to a bit or bit field; the opposite of *clear*. The term ‘set’ may also be used to generally describe the updating of a bit or bit field.

Set (*n*)**.** A subdivision of a *cache*. Cacheable data can be stored in a given location in one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache block* corresponding to that address was used least recently. See *Set-associative*.

Set-associative. Aspect of cache organization in which the cache space is divided into sections, called *sets*. The cache controller associates a particular main memory address with the contents of a particular set, or region, within the cache.

Slave. The device addressed by a master device. The slave is identified in the address tenure and is responsible for supplying or latching the requested data for the master during the data tenure.

Snooping. Monitoring addresses driven by a bus master to detect the need for coherency actions.

Snoop push. Response to a snooped transaction that hits a modified cache block. The cache block is written to memory and made available to the snooping device.

Stall. An occurrence when an instruction cannot proceed to the next stage.

Sticky bit. A bit that when *set* must be cleared explicitly.

Superscalar machine. A machine that can issue multiple instructions concurrently from a conventional linear instruction stream.

Supervisor mode. The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.

Synchronization. A process to ensure that operations occur strictly *in order*. See *Context synchronization*.

System memory. The physical memory available to a processor.

T

Tenure. The period of bus mastership. There can be separate address bus tenures and data bus tenures.

Throughput. The measure of the number of instructions that are processed per clock cycle.

Time-division multiplex (TDM). A single serial channel used by several channels taking turns.

Transaction. A complete exchange between two bus devices. A transaction is typically comprised of an address tenure and one or more data tenures, which may overlap or occur separately from the address tenure. A transaction may be minimally comprised of an address tenure only.

Transfer termination. Signal that refers to both signals that acknowledge the transfer of individual beats (of both single-beat transfer and individual beats of a burst transfer) and to signals that mark the end of the tenure.

Translation lookaside buffer (TLB). A cache that holds recently-used *page table entries*.

U

User mode. The operating state of a processor used typically by application software. In user mode, software can access only certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.

V

Virtual address. An intermediate address used in the translation of an *effective address* to a physical address.

Virtual memory. The address space created using the memory management facilities of the processor. Program access to *virtual memory* is possible only when it coincides with *physical memory*.

W

Way. A location in the cache that holds a cache block, its tags, and status bits.

Word. A 32-bit data element.

Write-back. A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly, for example, when a modified cache block is *cast out* to make room for newer data.

Write-through. A cache memory update policy in which all processor write cycles are written to both the cache and memory.

Index 1
Register Index (Memory-Mapped Registers)

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Index 2
General Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

Z

Index 3
CPM Index

A

B

C

G

H

I

L

M

N

O

P

Q

R

S

T

U