MPC8323E PowerQUICC™ II Pro Integrated Communications Processor Reference Manual

Supports

MPC8323E

MPC8323

MPC8321E

MPC8321

MPC8323ERM
Rev. 2
10/2008

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

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

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

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: MPC8323ERM
Rev. 2, 10/2008

Contents

Contents

Contents

Contents

Chapter 5
System Configuration

Contents

Contents

Contents

Chapter 6
Arbiter and Bus Monitor

Contents

Chapter 7
e300 Processor Core Overview

Contents

Chapter 8
Integrated Programmable Interrupt Controller (IPIC)

Contents

Contents

Contents

Chapter 10
Local Bus Controller

Contents

Contents

Contents

Chapter 13
PCI Bus Interface

Contents

Contents

Contents

Contents

Contents

Chapter 15
I^2C Interface

Contents

Chapter 16
DUART

Contents

Chapter 17
JTAG/Testing Support

Chapter 18
System Interface

Contents

Contents

Chapter 19
Configuration

Contents

Contents

Chapter 21
Serial Peripheral Interface (SPI)

Contents

Contents

Chapter 24
UART Mode and Asynchronous HDLC

Contents

Chapter 25
BISYNC Mode

Contents

Chapter 26
UCC for Fast Protocols

Contents

Contents

Chapter 29
UCC Ethernet Controller (UEC)

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 31
UTOPIA L2 Bus Controller (UPC)

Contents

Chapter 32
Serial Interface with Time-Slot Assigner

Contents

Chapter 33
Serial ATM Microcode

Contents

Chapter 34
QUICC Multi-Channel Controller (QMC)

Contents

Contents

Contents

Contents

Chapter 36
Universal Serial Bus Controller

Contents

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

About This Book

This reference manual defines the functionality of the MPC8323E. It is written from the perspective of the MPC8323E, which is the superset device, and unless otherwise noted, the information applies also to the MPC8323, MPC8321E, and MPC8321. Note that the MPC8323 and MPC8321 do not support a security engine.

The MPC8323E is a cost-effective, highly integrated communications processor that addresses the requirements of several networking applications, including routers, industrial control, and test and measurement applications. The MPC8323E contains an embedded PowerPC™ core. The e300c2 processor core, with its 16 Kbytes of instruction and data cache, implements the PowerPC user instruction set architecture and provides hardware and software debugging support. This configuration of the core does not support an FPU. The MPC8323E extends the PowerQUICC™ I family, adding higher CPU performance, additional functionality, and faster interfaces while addressing the requirements related to time-to-market, price, power consumption, and package size.

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:

- Chapter 1, “Overview,” provides a high-level description of features and functionality of the MPC8323E integrated communications processor. It describes the MPC8323E, its interfaces, and the programming model. The functional operation of the MPC8323E, with emphasis on peripheral functions, is also described.
- Chapter 2, “Memory Map,” describes the memory map of the MPC8323E. An overview of the local address map is provided. Next, a complete listing of all memory-mapped registers is provided, 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, their functional blocks, and I/O states. Also, these signals are listed by alphabetical order.
- 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 MPC8323E.
- Chapter 5, “System Configuration,” provides an overview of several functions that control the local access windows, system configuration, software watchdog, real time clock, periodic and general purpose timers, power management, protection, and general utilities.

- Chapter 6, “Arbiter and Bus Monitor,” provides an overview of the arbiter in the MPC8323E device. Also, it describes configuration, control, and status registers of the arbiter.
- Chapter 7, “e300 Processor Core Overview,” provides an overview of the basic functionality of the processor core and briefly describes how the functional units interact.
- Chapter 8, “Integrated Programmable Interrupt Controller (IPIC),” describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. It also provides a definition of the external interrupt signals and their functions. In addition, the interrupt configuration, control, and status registers are described in this chapter.
- Chapter 9, “DDR Memory Controller,” describes the 32-bit DDR SDRAM memory controller of the MPC8323E.
- Chapter 10, “Local Bus Controller,” describes the local bus controller (LBC) of the MPC8323E. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM) and user-programmable machines (UPMs) of the LBC. Also, it includes an initialization and applications information section with many specific examples of its use.
- Chapter 11, “Sequencer,” describes how the I/O sequencer (IOS) switches transactions among its ports, using a buffer pool to minimize blocking. It also provides address translation on outbound PCI transactions.
- Chapter 12, “DMA/Messaging Unit,” describes the four-channel high speed general-purpose DMA controller of the MPC8323E. The channels share buffer space in the IOS to facilitate the gathering and sending of data. The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This communication unit operates with generic messages and doorbell registers. This block also provides a DMA controller that transfers blocks of data independent of the local processor or PCI hosts.
- Chapter 13, “PCI Bus Interface,” describes the PCI interface, which complies with the *PCI Local Bus Specification*, Rev. 2.3. This chapter provides a basic description of PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification.
- Chapter 14, “Security Engine (SEC) 2.2,” describes the SEC 2.2 that is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the e300 core of the MPC8323E. It is optimized to process all the algorithms associated with IPSec. The SEC 2.2 (implemented in the MPC8323E) is derived from integrated security cores found in other members of the PowerQUICC family, including SEC 1.0, the version implemented in the MPC8272/MPC8248. Note that the MPC8323 and MPC8321 do not support a security engine.
- Chapter 15, “$I^{2C\ Interface}$,” describes the inter-IC (IIC or I^2C) bus controller of the device. 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 MPC8323E powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.

- Chapter 16, “DUART,” describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- Chapter 17, “JTAG/Testing Support,” describes the joint test action group (JTAG) interface of the device to facilitate boundary-scan testing. The JTAG interface complies to the IEEE Std 1149.1™ boundary-scan specification.
- Chapter 18, “System Interface,” describes the QUICC Engine system interface, which consists of functions that interface to the coherent system bus and the CPU. The system interface includes the serial DMA (SDMA), which transfers data from the QUICC Engine module to memory, and the interrupt controller, which is accessed by the CPU to verify the cause of an interrupt from the QUICC Engine module.
- Chapter 19, “Configuration,” details the QUICC Engine control registers, which allow the CPU to control and monitor the operation of the RISC controllers in the QUICC Engine block. These registers are used to configure certain global options and to create specific commands related to the communication protocols.
- Chapter 20, “Multiplexing and Timers,” describes the QUICC Engine multiplexing and timers logic (CMX), which routes clocks and connects the physical interfaces (such as modem lines, TDM lines and proprietary serial lines) to the QUICC Engine peripherals (UCCs, UPC, TDMs, etc.)
- Chapter 21, “Serial Peripheral Interface (SPI),” provides a description of the serial peripheral interface (SPI), which allows the exchange of data with other devices containing an SPI, as well as with the Ethernet PHY for configuration, and peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices. There are two different SPIs, one that operates as a normal SPI, and the other that is dedicated for the use of MIIMCOM. The SPI can operate in QUICC Engine mode or in CPU mode.
- Chapter 22, “Unified Communications Controllers (UCCs),” provides a general overview of the UCCs, the set of the protocols for them, and the common UCC programming model.
- Chapter 23, “UCC for Slow Protocols,” describes the UCC programmed as a slow communication controller, used for UART, BISYNC, Multichannel HDLC (QMC) protocols, or Serial ATM (SAM). When configuring a UCC to one of these protocols, read this chapter first and then proceed to the protocol specific chapter:
- Chapter 24, “UART Mode and Asynchronous HDLC,” outlines the universal asynchronous receiver transmitter (UART) protocol, commonly used to send low-speed data between devices through asynchronous links. The UART is also used as a local port to run board debugger software when synchronous communications are required.
- Chapter 25, “BISYNC Mode,” describes the UCC configured as a BISYNC controller that can handle basic BISYNC protocol in normal and transparent modes. This chapter discusses the three classes of BISYNC frames: transparent, nontransparent with header, and nontransparent without header. The controller can work with the time-slot assigner (TSA) or with the non-multiplexed serial interface (NMSI), and it has separate transmit and receive sections whose operations are asynchronous with the CPU, and either synchronous or asynchronous with other UCCs.
- Chapter 26, “UCC for Fast Protocols,” provides a general overview of the UCC when used for fast protocols and the common programming model. The fast protocols include ATM over

UTOPIA L2, and high-speed serials (Ethernet, HDLC, HDLC bus, Transparent). Chapter 22, "Unified Communications Controllers (UCCs)," should be read before reading this chapter.

- Chapter 27, "HDLC Controller," addresses high level data link control (HDLC), which is one of the most common protocols in layer 2 of the seven-layer OSI model—the data link layer (DLL). 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.
- Chapter 28, "Transparent Controller," describes the UCC transparent controller, which functions as a high-speed serial-to-parallel and parallel-to-serial converter. Transparent mode provides a clear channel on which the UCC performs no bit-level manipulation.
- Chapter 29, "UCC Ethernet Controller (UEC)," describes the Ethernet IEEE Std 802.3™ protocol, which is a widely-used LAN based on the carrier-sense multiple access/collision detect (CSMA/CD) approach.
- Chapter 30, "ATM Controller AAL0 and AAL5," describes the ATM controller, which 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. The ATM controller performs segmentation and reassembly (SAR) functions of AAL5 and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.
- Chapter 31, "UTOPIA L2 Bus Controller (UPC)," covers the UPC (UTOPIA L2 bus controller). The QUICC Engine block supports UTOPIA level 2 for both master and slave modes.
- Chapter 32, "Serial Interface with Time-Slot Assigner," pertains to the serial interface (SI), which manages the routing of four TDM lines to the QUICC Engine block serial drivers, the UCCs, for receive and transmit. The time-slot assigner (TSA) supports the serial bus rate for most standard TDM buses, including T1 and CEPT highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates. Each TDM can support E3 or DS-3 rates as a clear channel in either a parallel-nibble or serial interface.
- Chapter 33, "Serial ATM Microcode," addresses the serial ATM microcode (SAM), which complies with the ITU-T I.432 (transmission convergence sublayer) for SDH-based ATM systems.
- Chapter 34, "QUICC Multi-Channel Controller (QMC)," pertains to the QUICC multi-channel controller (QMC) functionality, which can emulate up to 64 time-division serial channels, using a single unified communication controller (UCC), and a time-division-multiplexed (TDM) physical interface.
- Chapter 35, "Inverse Multiplexing for ATM (IMA)," describes the ATM functionality provided through an implementation of inverse multiplexing for ATM (IMA). This chapter provides a broad overview of the IMA specifications and the specific implementation.
- Chapter 36, "Universal Serial Bus Controller," describes the USB controller, which provides communication with other devices via a USB connection. This chapter describes the QUICC Engine USB controller, including basic operation, the parameter RAM, and registers.
- Appendix A, "Revision History," lists the major differences between revisions of this 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 PowerPC architecture and computer architecture in general:

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

Related Documentation

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

- *e300 Core Reference Manual*—This book provides a more detailed description of the e300 core.
- 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 devices, an addendum may be 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.
- Product briefs—Each device has a product brief that provides an overview of its features.
- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale processors.

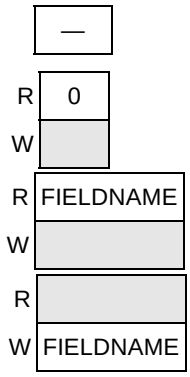
Additional literature is published as new processors become available. For a current list of documentation, refer to http://www.freescale.com.

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 a 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
AESU	Advanced encryption standard unit
AFEU	ARC four execution unit
BD	Buffer descriptor
BIST	Built-in self test
CD	Collision detect
COL	Collision
CPM	Communication processor module
CRC	Cyclic redundancy check
CRS	Carrier sense
CSB	Coherent system bus
CSMA	Carrier-sense multiple access
DDR	Double data rate
DEU	Data encryption standard execution unit
DMA	Direct memory access
DRAM	Dynamic random access memory
DTLB	Data translation lookaside buffers
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
EHCI	Enhanced host controller interface
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FS	Full-speed
FCS	Frame-check sequence
GMII	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
GTM	General purpose timers
IAD	Internet access device
I^2C	Inter-integrated circuit
IEEE	Institute of Electrical and Electronics Engineers
IOS	I/O sequencer

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
IPG	Interpacket gap
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
LALE	LBC external address latch enable
LBC	Local bus controller
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MCP	Machine-check interrupt
MDI	Medium-dependent interface
MDEU	Message digest execution unit
MIB	Management information base
MII	Media independent interface
MMU	Memory management unit
MPH	Multi-port host
MSB	Most-significant byte
msb	Most-significant bit
OSI	Open systems interconnection
PCI	Peripheral component interconnect
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PIT	Periodic interval timer
PKEU	Public key execution unit
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

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
RMON	Remote monitoring
RMW	Read-modify-write
RNG	Random number generator
RTBI	Reduced ten-bit interface
RTC	Real time clock module
Rx	Receive
RxBD	Receive buffer descriptor
SCL	Serial clock
SDA	Serial data
SFD	Start frame delimiter
SI	Serial interface
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TBI	Ten-bit interface
TLB	Translation lookaside buffer
TSEC	Three-speed Ethernet controller
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
ULPI	USB low-pin count interface
UPM	User-programmable machine
USB	Universal serial bus
UTMI	USB transceiver macrocell interface
UTP	Unshielded twisted pair
WDT	Watchdog timer
ZBT	Zero bus turnaround

Chapter 1
Overview

This chapter provides an overview of the MPC8323E and MPC8321E PowerQUICC™ II Pro processor features, including a block diagram showing the major functional components. The MPC8323E is a cost-effective, highly integrated communications processor that addresses the requirements of several networking applications, including routers, industrial control, and test and measurement applications. The MPC8323E extends the PowerQUICC™ family, adding higher CPU performance, additional functionality, and faster interfaces while addressing the requirements related to time-to-market, price, power consumption, and package size. This manual is written from the perspective of the MPC8323E, and unless otherwise noted, the information applies also to the MPC8323, MPC8321E, and MPC8321. Note that the MPC8323 and MPC8321 do not support a security engine.

1.1 MPC8323E PowerQUICC II Pro Processor Overview

Figure 1-1 shows the major functional units within the MPC8323E. One major component of the MPC8323E is the e300c2 core, which includes 16 Kbytes of instruction and data caches and is fully compatible with the PowerPC™ user instruction set. Another is the QUICC Engine™ 1.0 communications complex, which provides termination, interworking and switching between a wide range of protocols including ATM, Ethernet, TDM, and HDLC. Note that the MPC8321E and MPC8321 do not run ATM on the Utopia interface. Other major features include a DDR1/DDR2 SDRAM memory controller, a 32-bit PCI 2.3 controller, a flexible local bus, and a dedicated security engine.

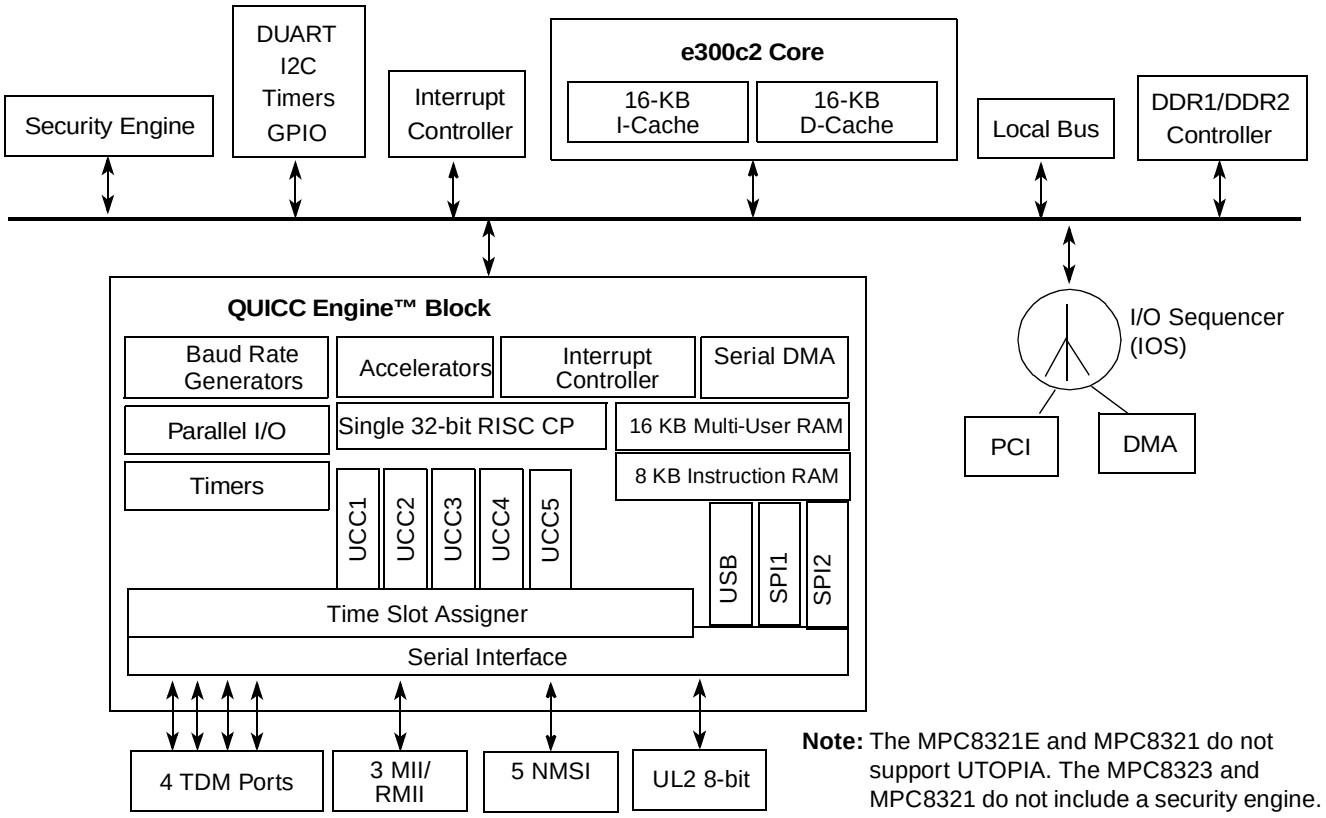


Figure 1-1. MPC8323E Block Diagram

The major features of this device are as follows:

- e300c2 Power Architecture™ processor core
  - — High-performance, superscalar processor core with a four-stage pipeline and low interrupt latency times
  - — Dual integer units, load/store, system register, and branch processing units
  - — 16-Kbyte instruction cache and 16-Kbyte data cache with lockable capabilities
  - — Dynamic power management
  - — Enhanced hardware program debug features
  - — Software-compatible with Freescale processor families implementing Power Architecture technology
  - — Separate PLL that is clocked by the system bus clock
- QUICC Engine 1.0 block
  - — 32-bit RISC controller for flexible support of the communications peripherals with the following features:
    - – One clock per instruction
    - – Separate PLL for operating frequency that is independent of system's bus and core frequency for power and performance optimization

  - 32-bit instruction object code
  - Executes code from internal ROM or RAM
  - 32-bit arithmetic logic unit (ALU) data path
  - Modular architecture allowing for easy functional enhancements
  - Slave bus for CPU access of registers and multiuser RAM space
  - 8 Kbytes of instruction RAM
  - 16 Kbytes of multiuser data RAM
- Serial DMA channel for receive and transmit on all serial channels
- Five unified communication controllers (UCCs) supporting the following protocols and interfaces:
  - 10/100 Mbps Ethernet/IEEE 802.3 through MII and RMII interfaces.
  - ATM through the UPC (MPC8323-specific)
  - Serial ATM through the serial interface
  - HDLC/Transparent (bit rate up to 70 Mbps)
  - HDLC BUS (bit rate up to 10 Mbps)
  - Asynchronous HDLC (bit rate up to 2 Mbps)
  - UART
  - BISYNC (bit rate up to 2 Mbps)
  - Four TDM interfaces supporting up to 64 QUICC multichannel controller (QMC) channels, each running at 64 Kbps
- ATM controller
  - Full duplex SAR protocols at OC-3 rate through UTOPIA L2 (MPC8323-specific)
  - AAL5, AAL0 protocols. TM 4.1 CBR, VBR, UBR, UBR+ traffic types
  - 64 K external connections
  - Inverse multiplexing ATM capability (IMA)
  - 1 UTOPIA L2 bus controllers (UPC) supporting 31 ports (MPC8323-specific)
- Universal serial bus (USB) controller
  - USB 1.1 full/low rate compatible
  - USB 2.0 full/low rate compatible (not high speed)
  - USB host mode
  - USB slave mode
- Two serial peripheral interfaces (SPIs). SPI2 is dedicated to Ethernet PHY management.
- Time slot assigner and 4 TDM serial interfaces
  - Independent Rx and Tx routing RAM with 512 routing entries each.
  - Time slot assigner with bit or byte resolution.
- 13 independent baud rate generators and 19 input pins to clock the UCCs, SI, UPC, USB, time-stamps and timer.
- Four independent 16-bit timers that can be interconnected as two 32-bit timers

- Security engine optimized to handle all the algorithms associated with IPSec, SSL/TLS, SRTP, IEEE Std 802.11i™, iSCSI, and IKE processing. The security engine contains one crypto-channel, a controller, and a set of crypto execution units (EUs). The execution units are:
  - Data encryption standard execution unit (DEU)
    - DES and 3DES algorithms
    - Two key (K1, K2) or three key (K1, K2, K3) for 3DES
    - ECB and CBC modes for both DES and 3DES
  - Advanced encryption standard unit (AESU)
    - Implements the Rijndael symmetric-key cipher
    - Key lengths of 128-, 192-, and 256-bits
    - ECB, CBC, CCM, and counter (CTR) modes
  - Message digest execution unit (MDEU)
    - SHA with 160- or 256-bit message digest
    - MD5 with 128-bit message digest
    - HMAC with either algorithm
  - One crypto-channel supporting multi-command descriptor chains
    - 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
    - Support for multiple outstanding bus transactions
    - Scatter/gather capability
    - Fetch FIFOs in the crypto-channel
- DDR SDRAM memory controller
  - Programmable timing supporting both DDR1 and DDR2 SDRAM
  - 32-bit data interface, up to 266-MHz data rate
  - 512-Mbyte addressable space
  - Support for 2 x16 devices
  - The following SDRAM configurations are supported:
    - 64-Mbit to 1-Gbit devices with x8/x16/x32 data ports (no direct x4 support). Some 2-Gbit devices are supported depending on the internal device configuration.
  - Single physical bank of memory (one chip select) with eight logical banks
  - Support for up to 8 simultaneous open pages
  - Sleep-mode support for SDRAM self refresh
  - Supports auto refresh
  - On-the-fly power management using CKE
  - Registered SIMM support
  - 2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL_18 compatible I/O for DDR2

- PCI interface
  - Designed to comply with *PCI Local Bus Specification, Revision 2.3*
  - 32-bit PCI interface operating at up to 66 MHz
  - PCI 3.3-V compatible
  - Support for host and agent modes
  - Memory prefetching of PCI read accesses and support for delayed read transactions
  - Inbound and outbound write posting
  - Internal configuration registers accessible from PCI
  - Selectable hardware-enforced coherency
- Local bus controller (LBC)
  - Multiplexed and non-multiplexed 26-bit address and 16-bit data operating at up to 66 MHz
  - Four chip selects supporting four external slaves
  - Supports up to 64 Mbytes of memory
  - Up to eight-beat burst transfers
  - 16- and 8-bit ports
  - Two protocol engines available on a per chip select basis:
    - General-purpose chip select machine (GPCM)
    - Three user programmable machines (UPMs)
- Integrated programmable interrupt controller (IPIC)
  - Functional and programming compatibility with the MPC8349 interrupt controller
  - Support for external and internal discrete interrupt sources
  - Support for one external (optional) and seven internal machine check interrupt sources
  - Programmable highest priority request
  - Four groups of interrupts with programmable priority
  - External and internal interrupts directed to communication processor
  - Redirects interrupts to external $\overline{\text{PCI\_INTA}}$ signal when in core disable mode
  - Unique vector number for each interrupt source
- I^2C interface
  - Two-wire interface
  - Multiple-master support
  - Master or slave I^2C mode support
  - On-chip digital filtering rejects spikes on the bus
- DMA (Direct memory access) controller
  - Four independent fully programmable DMA channels
  - Misaligned transfer capability for source/destination address
  - Data chaining and direct mode
  - Interrupt on completed segment and chain

- DUART
  - Two 4-wire interfaces (RxD, TxD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$)
  - Programming model compatible with the original 16450 UART and the PC16550D
- Parallel I/O
  - General-purpose I/O (GPIO)
    - 128 parallel I/O pins multiplexed on various chip interfaces
  - Open drain capability
  - Interrupt capability
- System timers
  - Periodic interrupt timer
  - Real-time clock
  - Software watchdog timer
  - Two general-purpose timers
- IEEE 1149.1-compatible JTAG boundary scan
- Integrated PCI bus and SDRAM clock generation

1.2 MPC8323E Architecture Overview

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

1.2.1 Power Architecture Core

The device contains the e300c2 Power Architecture processor core, which is an enhanced version of the MPC603e core (used in previous generations of PowerQUICC II processors). Enhancements include integrated parity checking, dual integer units, and other performance-enhancing features. The e300 core is upward software-compatible with existing MPC603e core-based products.

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

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

The e300 core is a low-power implementation of the family of microprocessors that implements Power Architecture technology. The core implements the 32-bit portion of the architecture, which provides 32-bit effective addresses and integer data types of 8, 16, and 32 bits. Note that the e300c2 core does not support floating-point operations.

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

The e300c2 core integrates five execution units—two integer units (IU1 and IU2) with full multiply and divides, a branch processing unit (BPU) with static branch prediction, a load/store unit (LSU) for data transfers, and a system register unit (SRU). The ability to execute five instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput. Most integer instructions execute in one clock cycle; two integer instructions may be executed at the same time with the dual integer units.

The e300c2 core provides independent on-chip, 16-Kbyte, eight-way set-associative, physically-addressed instruction and data caches with parity and integrated way lock capabilities. The processor also features independent on-chip instruction and data memory management units (MMUs). The MMUs contain 64-entry, two-way set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged virtual memory address translation. The caches use a pseudo least recently used (PLRU) replacement algorithm; the TLBs use a least recently used (LRU) replacement algorithm. The processor also supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays of eight entries each. Effective addresses are compared simultaneously with all eight entries in the BAT array during block translation. In accordance with the architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

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

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

Figure 1-2 provides a block diagram of the e300 core that shows how the execution units (IU1, IU2, BPU, LSU, and SRU) operate independently and in parallel. Note that this is a conceptual diagram and does not attempt to show how these features are physically implemented on the chip.

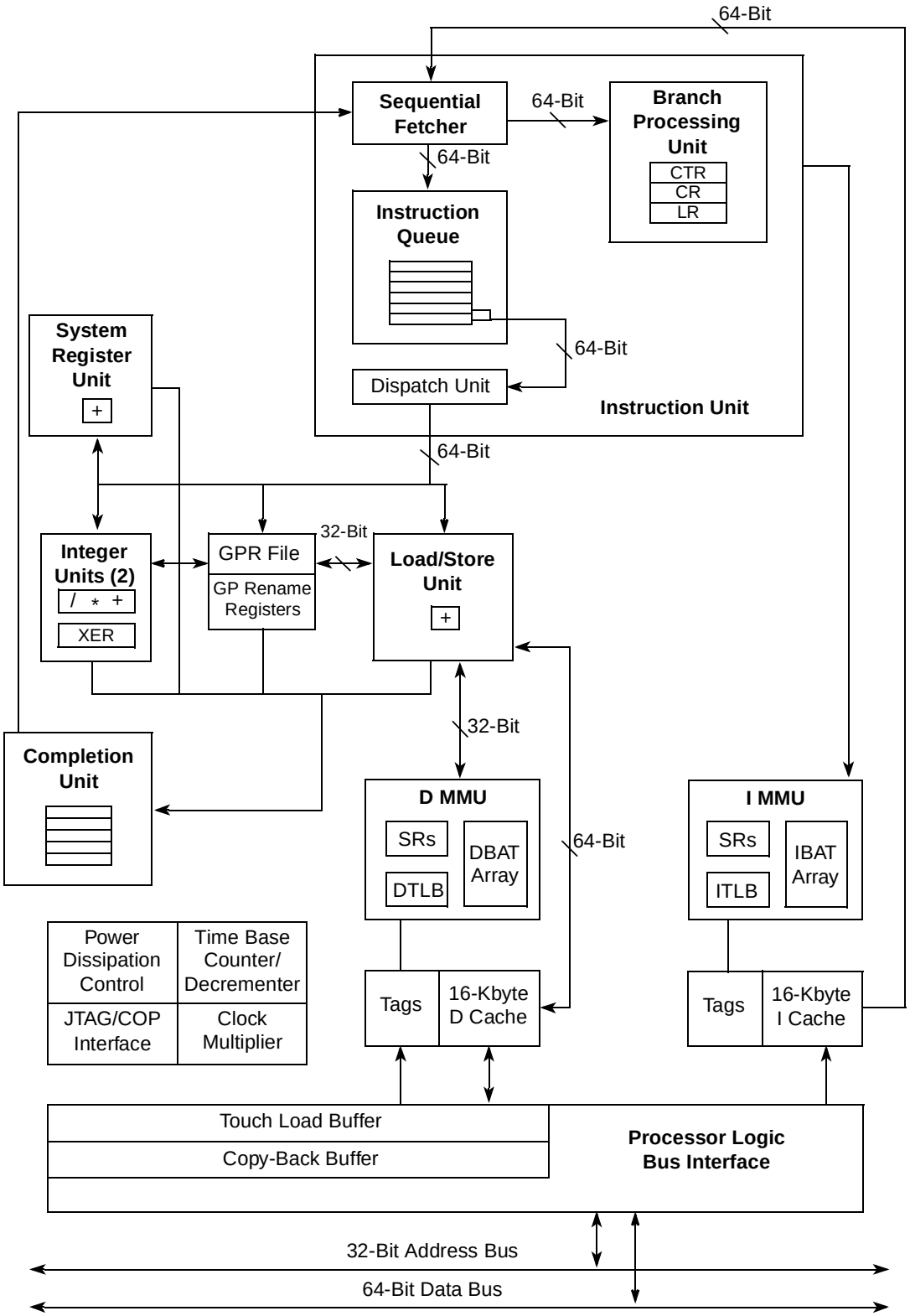


Figure 1-2. MPC8323E Integrated e300c2 Core Block Diagram

1.2.2 QUICC Engine 1.0 Block

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

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

- Five unified communication controllers (UCCs) with the following features:
  - Ethernet, ATM, HDLC/HDLC bus and transparent protocols (also known as fast protocols).
  - Serial ATM and QMC that are user compatible with the SCC of the CPM (also known as slow protocols)
  - Ethernet for the First Mile (IEEE 802.3ah 2BASE-TL and 10PASS-TS)
  - UART, BISYNC and Async HDLC that are user compatible with the SCC of the CPM, require a software patch.
  - The HDLC and transparent protocols are user compatible with the FCC of the CPM.
- One UTOPIA L2 controller (UPC)[1] for 31 ports (MPC8323-specific)
- Two serial peripheral controllers (SPI1 and SPI2[2])
- Time slot assigner and serial interface (SI) for 4 TDMs and full duplex routing RAM of 512 entries.
- One universal serial bus controller (USB 1.1/2.0)

The UCCs are similar to the PowerQUICC II peripherals: SCC (BISYNC, UART, and HDLC bus), and FCC (fast Ethernet, HDLC, transparent, and ATM). In addition, 31 UTOPIA PHYs are supported in ATM mode. Note that the MPC8321E does not run ATM on Utopia.

1. POS protocol will not be supported in the QUICC Engine 1.0 block.
2. SPI2 can be used only for Ethernet management

Figure 1-3 shows the internal architecture and the interfaces provided by the QUICC Engine block. The QUICC Engine 1.0 block contains five UCCs controlled by a single RISC engine. A common multiuser RAM is used to store parameters for RISC engine. The RISC has a ROM associated with it, which contains the code image. The instruction RAM is used to run RISC code from the RAM.

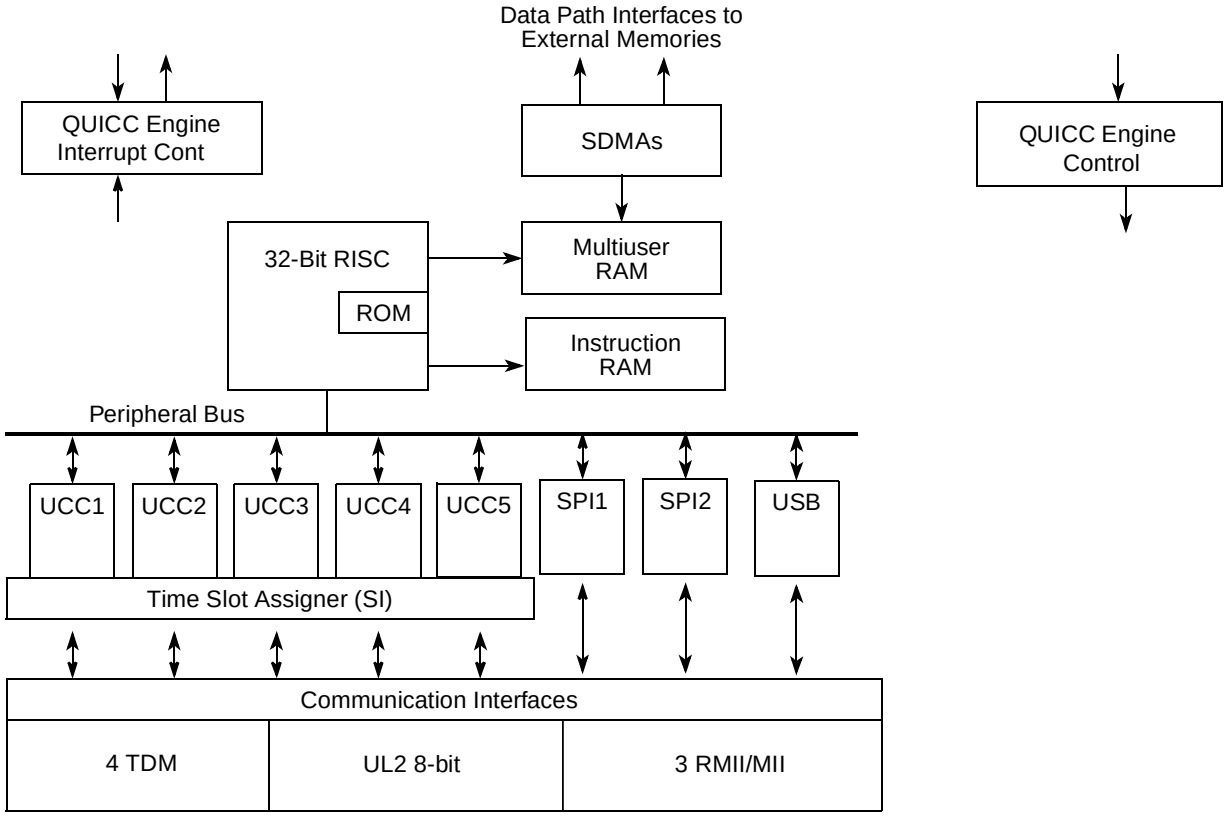


Figure 1-3. QUICC Engine 1.0 Block Architectural Block Diagram

1.2.2.1 QUICC Engine Interfaces

The following sections describe the different QUICC Engine interfaces.

1.2.2.1.1 System CPU Interface

The QUICC Engine block communicates with the CPU controller core in the following ways:

- Many parameters are exchanged through the multiuser RAM.
- The QUICC Engine block can execute special commands issued by the CPU. These commands should only be issued in special situations such as exceptions and error recovery.
- The QUICC Engine block generates interrupts through the SI (system interface) interrupt controller.
- The CPU can read/clear the QUICC Engine block status/event registers at any time.

1.2.2.1.2 QUICC Engine Communication Interfaces

The following are the total number of interfaces in the QUICC Engine block:

- 8-bit level 2 UTOPIA (MPC8323E-specific)
- 3 MII or RMII
- 4 TDMs, each containing data, clocks, sync, and strobes

Some of the interfaces are multiplexed and cannot be used simultaneously.

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

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

- SCC DPLL is not provided
- SCC 10 Base T (7-wire Ethernet) is no longer provided
- HDLC bus protocol programming model is FCC- instead of SCC-compatible
- IDMA I^2C functions are provided as part of the SIU.
- Support for policer features as standard on the QUICC Engine block
- Enhanced Ethernet controller which provides support for frame filtering based on the VLAN tag or any Ethernet type field and parsing of frame headers to perform table lookups
- ATM available bit rate (ABR) scheduling mode is not supported. Other scheduling modes are supported.
- UTOPIA external rate is supported, but the QUICC Engine block does not transmit idle cells when no data is available. (MPC8323-specific)
- GCI circuits through the serial interface are not supported.
- The instruction RAM is indirectly accessed.

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

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

- ATM enhancements:
  - Support for policer features as standard on the QUICC Engine block
  - Address look-up enhancements:
    - Internal mini-CAM
    - Lookup based on user’s defined cell extra header
  - Transmit scheduler enhancements:
    - Small memory foot-print scheduler for low bit-rate connections (scalable APC)
    - Hierarchal frame based scheduling
    - APC flux compensation
  - Simultaneous processing of multiple cells (multi-threading)
- Enhanced Ethernet features that provide for:
  - Frame filtering based on the MAC destination and source address, VLAN tag field and parsing of frame headers to perform table lookups.
  - IP support for IPv4 and IPv6 packets including TOS and header checksum processing.

  - UEC controller for 10/100 Mbps with support of VLAN
- USB protocol: automatic transmission of SOF tokens.
- SPI controller implement Ethernet MII serial management interface for up to 32 PHYs.
- Serial interface
  - Nibble-parallel data interface on all TDMs.
- TC layer for serial ATM supported as a microcode package.
- User can modify the peripheral's (UCC, SPI) parameter RAM base address in the multi-user RAM
- User programmable FIFO size for UCC fast protocols.
- The QUICC Engine operating frequency is independent of the SIU bus frequency, providing greater performance/power flexibility.
- Two independent time-stamp registers triggered by an external or internal clock

1.2.2.4 Software Migration from the MPC82xx/MPC85xx Family Devices

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

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

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

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

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

- Full compatibility with the QUICC multi-channel controller (QMC) running on top of the UCC providing 64 channels compatible with the MPC82xx QMC

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

The SPI and USB peripherals are software compatible with those of the CPM. Also, the QUICC Engine timers are compatible with the CPM's. Finally, serial ATM is offered as a standard microcode protocol for the UCC instead of the FCC2 UTOPIA and hardware enabled TC layer of the CPM.

1.2.2.5 Serial Protocol Table

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

Table 1-1. QUICC Engine 1.0 Protocols

Protocol	Controller				
	UCC	SPI	USB	UPC	SI
ATM, IMA (UTOPIA)	√	—	—	√	—
Serial ATM, IMA	√	—	—	—	√
Ethernet	√	—	—	—	—
HDLC	√	—	—	—	—
HDLC_BUS	√	—	—	—	—
Async HDLC	√[1]	—	—	—	—
BiSync	√[1]	—	—	—	—
Transparent	√	—	—	—	—
UART	√[1]	—	—	—	—
Multi channel HDLC/Transparent TDM	√	—	—	—	√
ISDN (IDL)	√	—	—	—	√
SPI	—	√	—	—	—
Ethernet management (SMI)	√ (Opt)	√	—	—	—
USB	—	—	√	—	—

1 Requires a S/W patch from Freescale

1.2.2.6 QUICC Engine UCC Capabilities

Not all the UCCs in the QUICC Engine block can be configured for all of the protocols. Table 1-2 lists the available protocols per UCC.

Table 1-2. UCC Protocol Enablement in the QUICC Engine Block

Protocol	Controller				
	UCC1	UCC2	UCC3	UCC4	UCC5
ATM	√	—	√	—	√
Ethernet	—	√	√	√	—
QMC/ Serial ATM	√	√	√	√	√
NMSI (HDLC/ Transparent/ BiSync/ UART/ Async HDLC/ ISDN)	√	√	√	√	√

1.2.2.7 QUICC Engine Configurations

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

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

1.2.3 Security Engine

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

The security engine supports DES, 3DES, MD-5, SHA-1, AES, and RC-4 encryption algorithms in hardware.

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

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

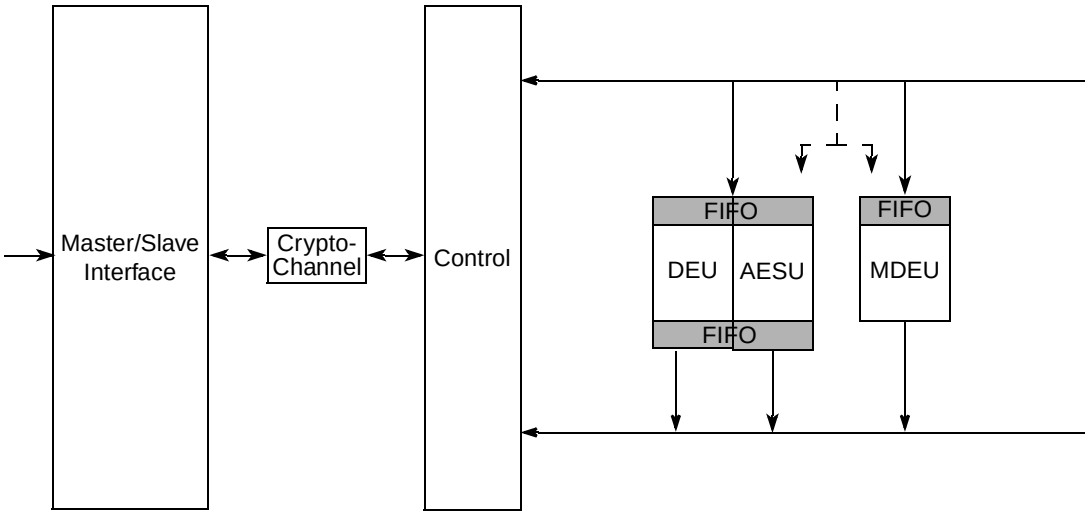


Figure 1-4. Integrated Security Engine Functional Blocks

1.2.4 DDR Memory Controller

This fully programmable DDR SDRAM controller supports most JEDEC standard x8 or x16 DDR1 or DDR2 memories available today, including buffered and unbuffered DRAM modules. However, mixing nonregistered and registered DRAM modules in the same system is not supported. Dynamic power management and auto-precharge modes simplify memory system design.

The DDR memory controller includes the following features:

- Support for DDR1 and DDR2 SDRAM
- 32-bit SDRAM data bus
- Programmable settings for meeting all SDRAM timing parameters
- Many different SDRAM configurations supported
  - Support for one physical bank (chip select)
  - Support for 64-Mbit to 1-Gbit devices with x8/x16/x32 data ports. Some 2-Gbit devices are supported depending on the internal device configuration.
  - Support for unbuffered and registered DRAM modules
- Support for data mask signals and read-modify-write operations for sub-double word writes
- Four-entry input request queue
- Open page management (dedicated entry for each sub-bank)
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management mode

1.2.5 PCI Controller

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

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

The device PCI controller includes the following distinctive features:

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

1.2.5.1 PCI Bus Arbitration Unit

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

- Supports three $\overline{\text{REQ}}$/$\overline{\text{GNT}}$ signal pairs, thus supporting three external masters. The device PCI controller is the fourth member of the arbitration pool.
- The bus arbitration unit allows fairness as well as a priority mechanism.
- A two-level round-robin scheme is used in which each device can be programmed within a pool of a high- or low-priority arbitration. One member of the low-priority pool is promoted to the high-priority pool. As soon as it is granted the bus, it returns to the low-priority pool.
- The unit can be disabled to allow a remote arbitration unit to be used.
- The unit can be isolated to allow power shut off of external devices.

1.2.6 Local Bus Controller (LBC)

The main component of the local bus controller (LBC) is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling four memory banks shared by 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 SRAM, EPROM, burstable RAM, and other peripherals. The LBC external address latch enable (LALE) signal allows multiplexing of addresses with data signals to reduce the device pin count.

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

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

- Memory controller with four memory banks (chip selects)
  - 32-bit address decoding with mask
  - Variable memory block sizes (32 Kbytes to 64 Mbytes in FCM mode)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - Atomic operation
- General-purpose chip-select machine (GPCM)
  - Compatible with SRAM, EPROM, and peripherals
  - Global (boot) chip-select available at system reset
  - Boot chip-select support for 8- and 16-bit devices
  - Minimum three-clock access to external devices
  - Two byte-write-enable signals ($\overline{\text{LWE}}$[0:1])
- Three user-programmable machines (UPMs)
  - Programmable-array-based machine controls external signal timing with a granularity of up to one quarter of an external bus clock period

— User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access
— User-specified control-signal patterns can be initiated by software
— Support for 8- and 16-bit devices
— Page mode support for successive transfers within a burst

1.2.7 Integrated Programmable Interrupt Controller (IPIC)

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

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

1.2.8 I^2C Interface

The inter-IC (IIC or I^2C) bus is a two-wire—serial data (SDA) and serial clock (SCL)— bidirectional serial bus that provides a simple, efficient method of data exchange between the system and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus minimizes the interconnections between devices. The synchronous, multi-master bus of the I^2C allows the connection of additional devices to the bus for expansion and system development.

The I^2C controller is a true multi-master bus which includes collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. The I^2C controller consists of a transmitter/receiver unit, clocking unit, and control unit. The I^2C unit supports general broadcast mode and on-chip filtering rejects spikes on the bus.

The I^2C interface includes the following features:

- Two-wire interface
- Multi-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave

- Calling address identification interrupt
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus
- Address broadcasting supported
- System initialization data is optionally loaded from I^2C EPROM by boot sequencer embedded hardware

1.2.9 DMA Controller

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

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct and extended chaining)
- Support for misaligned transfers
- Programmable bandwidth control between channels
- Interrupt on error and completed segment or chain

1.2.10 Dual Universal Asynchronous Receiver/Transmitter (DUART)

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

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

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

The DUART includes the following features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)

- Maskable transmit, receive, and line status interrupts
- Software-programmable baud rate generators that divide the system clock by 1 to (2^{16} – 1) and generate a 16x clock for the transmitter and receiver engines
- Clear to send ($\overline{\text{CTS}}$) and ready to send ($\overline{\text{RTS}}$) MODEM control functions
- Software-selectable serial-interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

1.2.11 System Timers

The system includes the following timers:

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

1.3 Application Examples

As technology standards for telecommunication and networking equipment including wireless infrastructure, DSLAMs, routers/switches, line cards, multi-channel modems, network storage, and IADs continue to change and new access technologies are introduced, a programmable communication processing platform that can evolve to accommodate such changes provides equipment vendors with a distinct competitive advantage. These system requirements allow the MPC8323E PowerQUICC II Pro™ processor to be used in a number of applications as described in the following sections.

1.3.1 SoHo Router

Figure 1-5 illustrates how a typical small office/home office (SoHo) router application can be realized with the MPC8323E.

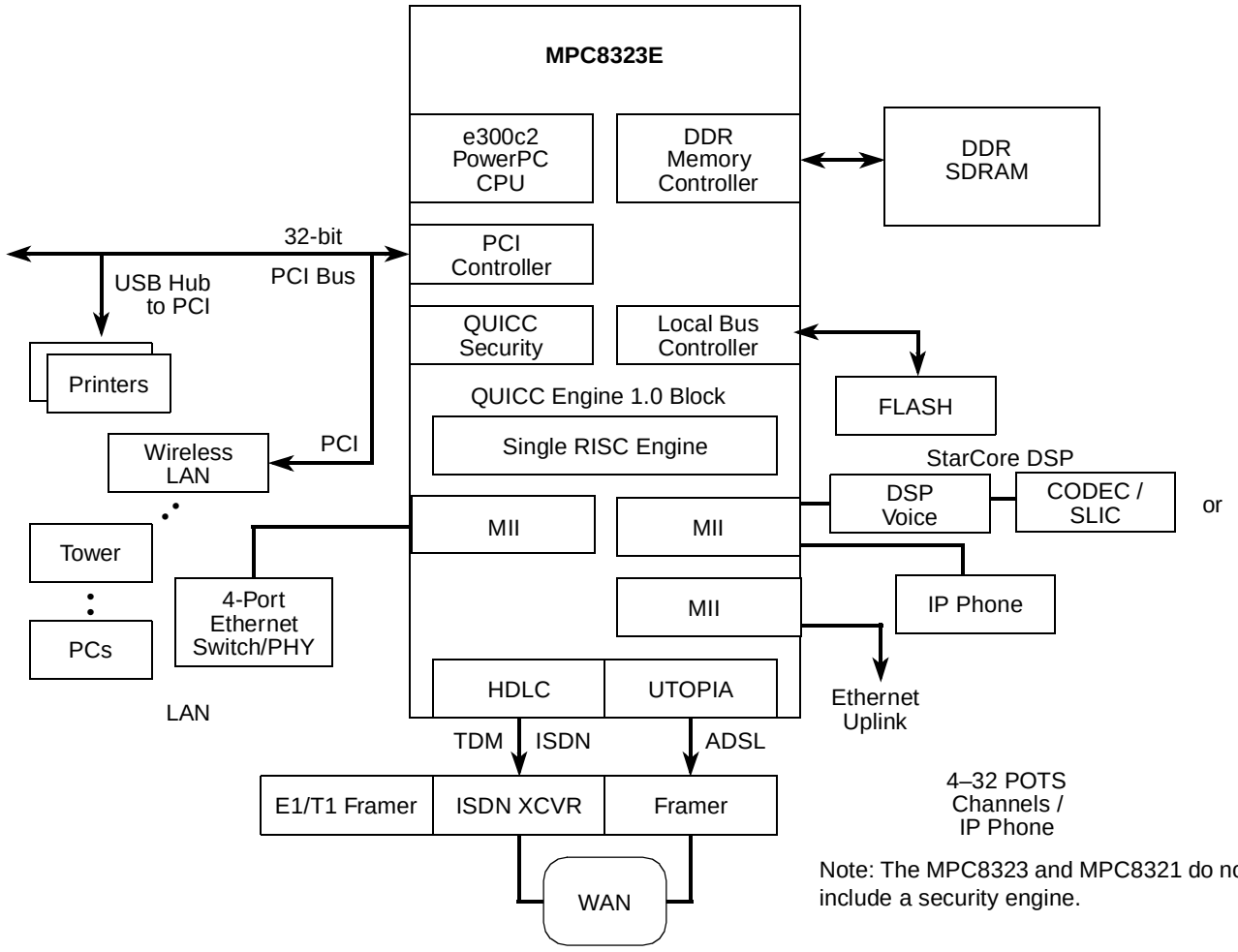


Figure 1-5. SoHo Router Using the MPC8323E

In this application the MPC8323E provides all of the processing, protocol, and interworking functions required to implement the SoHo router. Specifically, the QUICC Engine block is used to carry voice, data and video using IP over the LAN and WAN interfaces. On the LAN side, one UCC is used to connect to a 4-port fast Ethernet switch. One Ethernet interface is used for uplink, and one of the TDM interfaces is used to support HDLC, which provides a leased line E1/T1 connection or an ISDN connection. One UCC is used as an ATM interface supporting AAL5 cell sharing for dial-up ADSL connection, while the last UCC can be configured as serial (UART) or Ethernet (MII) for debug and control.

Alternatively, the remaining UCC of the MPC8323E could be used to support an Ethernet connection to a low cost digital signal processor (DSP) such as the Freescale StarCore family of DSPs based on the StarCore technology supporting 4 to 32 voice ports, which can be for plain old telephone system (POTS) telephones or for IP-based telephones using a combination of premium voice algorithms such as G.729a/b, G.723.1 or G.711. For very high-density voice ports, the MSC8122 DSP can be used through an Ethernet interface.

Other interfaces connected to the PCI bus can include a four-port universal serial bus (USB) hub for connecting equipment such as printers, copiers, scanners, and system back up disks. In addition, a wireless LAN interface can be connected to the PCI bus supporting 802.11-a/b/g/n connectivity within the office environment.

Finally, the security engine provides acceleration for encryption, authentication, and standards based tunnelling as required by IP-Sec.

1.3.2 DSLAM Line Card

The diagram in Figure 1-6 illustrates how an intelligent DSLAM line-card can be readily implemented.

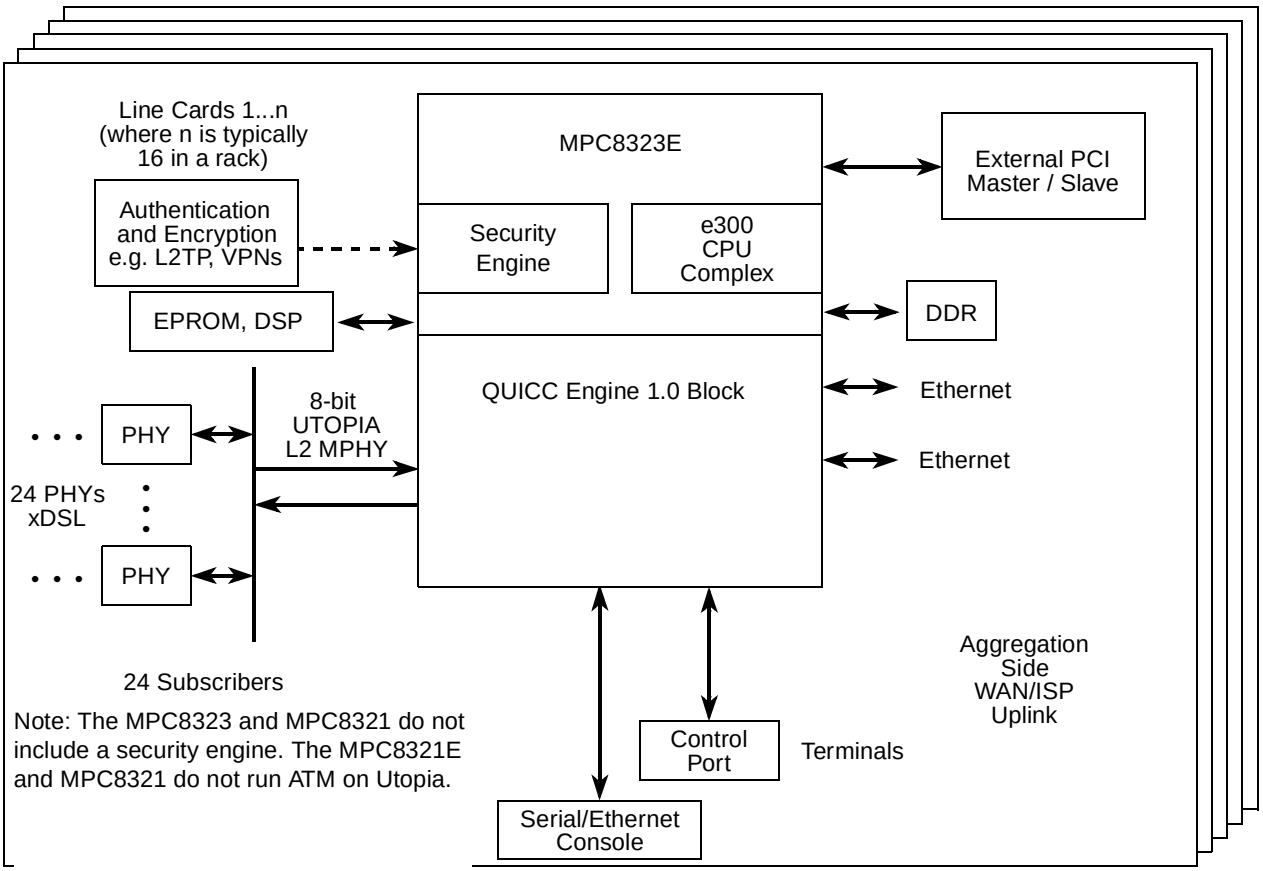


Figure 1-6. MPC8323E Line Card Implementation

In this example, the versatility of the UCCs enables the convergence of both packet and circuit switched networks because both ATM and Ethernet functions can be supported. As shown on the left-hand side of Figure 1-6, subscribers are connected to the DSLAM line-card through the DSL PHYs and the integrated UTOPIA interface on the MPC8323E. Note that the MPC8321 and MPC8321E do not run ATM on Utopia.

In this application, ATM-Ethernet interworking would transfer data from the ATM-based DSL subscriber inputs, terminating the ATM protocol within the QUICC Engine block while the e300c2 core performs the interworking tasks. The QUICC Engine block will then egress packets through the Ethernet protocol.

Chapter 2
Memory Map

This chapter describes the MPC8323E memory map. The internal memory mapped registers are described, including a complete listing of all memory mapped registers with cross references to the sections detailing descriptions of each.

2.1 Internal Memory Mapped Registers

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

2.2 Accessing IMMR Memory From the Local Processor

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

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

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

2.3 Complete IMMR Map

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

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

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

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

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

Table 2-1. IMMR Memory Map

Address	Use	Actual Size	Window	Cross Reference
0x00_0000–0x00_01FF	System configuration	512 bytes	512 bytes	Table 2-2
0x00_0200–0x00_02FF	Watchdog timer	16 bytes	256 bytes	Table 2-2
0x00_0300–0x00_03FF	Real time clock	32 bytes	256 bytes	Table 2-2
0x00_0400–0x00_04FF	Periodic interval timer	32 bytes	256 bytes	Table 2-2
0x00_0500–0x00_05FF	Global timers module 1	64 bytes	256 bytes	Table 2-2
0x00_0600–0x00_06FF	Global timers module 2	64 bytes	256 bytes	Table 2-2
0x00_0700–0x00_07FF	Integrated programable interrupt controller (IPIC)	128 bytes	256 bytes	Table 2-2
0x00_0800–0x00_08FF	System arbiter	30 bytes	256 bytes	Table 2-2
0x00_0900–0x00_09FF	Reset module	—	256 bytes	Table 2-2
0x00_0A00–0x00_0AFF	Clock module	44 bytes	256 bytes	Table 2-2
0x00_0B00–0x00_0BFF	Power management control module	—	256 bytes	Table 2-2
0x00_0C00–0x00_0CFF	QUICC Engine™ ports interrupts	24 bytes	256 bytes	Table 2-2,
0x00_0D00–0x00_0DFF	Reserved	—	256 bytes	—
0x00_0E00–0x00_0EFF	Reserved, should be cleared	—	256 bytes	—
0x0_0F00–0x0_0FFF	Reserved, should be cleared	—	256 bytes	—
0x00_1000–0x00_10FF	Clock control DDR	20 bytes	256 bytes	Table 2-2
0x00_1100–0x00_11FF	Clock Control, LBC	20 bytes	256 bytes	Table 2-2
0x00_1200–0x00_12FF	Reserved, should be cleared	—	256 bytes	—
0x00_1300–0x00_13FF	Reserved	—	256 bytes	—
0x00_1400–0x00_17FF	QUICC Engine parallel I/O ports	168 bytes	1 Kbyte	Table 2-2
0x00_1800–0x00_1BFF	Reserved	—	1 Kbyte	—
0x00_1C00–0x00_1FFF	Reserved	—	1 Kbyte	—
0x00_2000–0x00_2FFF	DDR MEMC	3.8 Kbytes	4 Kbytes	Table 2-2
0x00_3000–0x00_30FF	I^2C controller	24 bytes	256 bytes	Table 2-2
0x00_3100–0x0_03FFF	Reserved, should be cleared	—	3.5 Kbytes	—
0x00_4000–0x00_44FF	Reserved, should be cleared	—	—	—
0x00_4500–0x00_46FF	DUART	18 bytes x 2	4 Kbytes	Table 2-2
0x00_4700–0x00_4FFF	Reserved, should be cleared	—	—	—
0x00_5000–0x00_5FFF	LBC	224 bytes	4 Kbytes	Table 2-2
0x00_6000–0x00_7FFF	Reserved	—	—	—
0x00_8000–0x00_82FF	DMA	768 bytes	768 bytes	Table 2-2
0x00_8300–0x00_837F	PCI configuration	16 bytes	128 bytes	Table 2-2
0x0_8380–0x0_83FF	Reserved	16 bytes	128 bytes	—

Table 2-1. IMMR Memory Map (continued)

Address	Use	Actual Size	Window	Cross Reference
0x00_8400–0x00_84FF	IOS	256 bytes	256 bytes	Table 2-2
0x00_8500–0x00_85FF	PCI controller	128 bytes	256 bytes	Table 2-2
0x00_8600–0x02_5FFF	Reserved	—	—	—
0x02_6000–0x02_FFFF	Reserved, should be cleared	—	—	—
0x03_0000–0x03_FFFF	Security engine	52 Kbytes	64 Kbytes	Table 2-2
0x04_0000–0x0F_FFFF	Reserved, should be cleared	—	—	—
0x10_0000–0x1F_FFFF	QUICC Engine 1.0 block	256 Kbytes	1 Mbyte	Table 2-3, Table 2-4

Table 2-2 lists the memory-mapped registers.

Table 2-2. Memory Map

Offset	Register	Access	Reset	Section/Page
System Configuration Registers				
0x0_0000	Internal memory map base address register (IMMRBAR)	R/W	0xFF40_0000	5.2.4.1/5-6
0x0_0004	Reserved	—	—	—
0x0_0008	Alternate configuration base address register (ALTCBAR)	R/W	0x0000_0000	5.2.4.2/5-7
0x0_000C–0x0_001C	Reserved	—	—	—
0x0_0020	LBC local access window 0 base address register (LBLAWBAR0)	R/W	0x0000_0000[1]	5.2.4.3/5-8
0x0_0024	LBC local access window 0 attribute register (LBLAWAR0)	R/W	0x0000_0000[2]	5.2.4.4/5-9
0x0_0028	LBC local access window 1 base address register (LBLAWBAR1)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_002C	LBC local access window 1 attribute register (LBLAWAR1)	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0030	LBC local access window 2 base address register (LBLAWBAR2)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_0034	LBC local access window 2 attribute register (LBLAWAR2)	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0038	LBC local access window 3 base address register (LBLAWBAR3)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_003C	LBC local access window 3 attribute register (LBLAWAR3)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_0040–0x0_005C	Reserved	—	—	—
0x0_0060	PCI local access window 0 base address register (PCILAWBAR0)	R/W	0x0000_0000[3]	5.2.4.5/5-10
0x0_0064	PCI local access window 0 attribute register (PCILAWAR0)	R/W	0x0000_0000[4]	5.2.4.6/5-11
0x0_0068	PCI local access window 1 base address register (PCILAWBAR1)	R/W	0x0000_0000[5]	5.2.4.5/5-10
0x0_006C	PCI local access window 1 attribute register (PCILAWAR1)	R/W	0x0000_0000	5.2.4.6/5-11
0x0_0070–0x0_009C	Reserved	—	—	—
0x0_00A0	DDR local access window 0 base address register (DDRLAWBAR0)	R/W	0x0000_0000[6]	5.2.4.7/5-12
0x0_00A4	DDR local access window 0 attribute register (DDRLAWAR0)	R/W	0x0000_0000[7]	5.2.4.8/5-13

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_00A8	DDR local access window 1 base address register (DDRLAWBAR1)	R/W	0x0000_0000	5.2.4.7/5-12
0x0_00AC	DDR local access window 1 attribute register (DDRLAWAR1)	R/W	0x0000_0000	5.2.4.8/5-13
0x0_00B0–0x0_00FC	Reserved	—	—	—
Watchdog Timer (WDT) Registers				
0x0–0x3	Reserved	—	—	—
0x4	System watchdog control register (SWCRR)	R/W	0xFFFF_0003 or 0xFFFF_0007[8]	5.4.4.1/5-25
0x8	System watchdog count register (SWCNR)	R	0x0000_FFFF	5.4.4.2/5-26
0xC–0xD	Reserved	—	—	—
0xE	System watchdog service register (SWSRR)	R/W	0x0000	5.4.4.3/5-26
Real Time Clock Module Registers (RTC)				
0x00	Real time counter control register (RTCNR)	R/W	0x0000_0000	5.5.5.1/5-32
0x04	Real time counter load register (RTLDR)	R/W	0x0000_0000	5.5.5.2/5-33
0x08	Real time counter prescale register (RTPSR)	R/W	0x0000_0000	5.5.5.3/5-33
0x0C	Real time counter register (RTCTR)	R	0x0000_0000	5.5.5.4/5-34
0x10	Real time counter event register (RTEVR)	w1c	0x0000_0000	5.5.5.5/5-34
0x14	Real time counter alarm register (RTALR)	R/W	0xFFFF_FFFF	5.5.5.6/5-35
0x18–0x1F	Reserved	—	—	—
Periodic Interval Timer (PIT) Registers				
0x00	Periodic interval timer control register (PTCNR)	R/W	0x0000_0000	5.6.5.1/5-39
0x04	Periodic interval timer load register (PTLDR)	R/W	0x0000_0000	5.6.5.2/5-40
0x08	Periodic interval timer prescale register (PTPSR)	R/W	0x0000_0000	5.6.5.3/5-41
0x0C	Periodic interval timer counter register (PTCTR)	R	0x0000_0000	5.6.5.4/5-41
0x10	Periodic interval timer event register (PTEVR)	w1c	0x0000_0000	5.6.5.5/5-41
0x14–0x1F	Reserved	—	—	—
Global Timers Module				
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	5.7.5.1/5-49
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	5.7.5.1/5-49
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	5.7.5.2/5-52
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0xFFFF	5.7.5.3/5-53
0x16	Timer 2 global timers reference register (GTRFR2)			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x18	Timer 1 global timers capture register (GTCPR1)	R/W	0x0000	5.7.5.4/5-53
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	5.7.5.5/5-54
0x1E	Timer 2 global timers counter register (GTCNR2)			
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	5.7.5.2/5-52
0x22	Timer 4 global timers mode register (GTMDR4)			
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0xFFFF	5.7.5.3/5-53
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	5.7.5.4/5-53
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	5.7.5.5/5-54
0x2E	Timer 4 global timers counter register (GTCNR4)			
0x30	Timer 1 global timers event register (GTEVR1)	w1c	0x0000	5.7.5.6/5-54
0x32	Timer 2 global timers event register (GTEVR2)			
0x34	Timer 3 global timers event register (GTEVR3)			
0x36	Timer 4 global timers event register (GTEVR4)			
0x38	Timer 1 global timers prescale register (GTPSR1)	R/W	0x0003	5.7.5.7/5-55
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			
General Purpose (Global) Timer Module 2: All registers defined for GTM1 are also defined for GTM2; the base address of GTM2 registers is 0x0_06*nn*.				
Integrated Programmable Interrupt Controller (IPIC)				
0x00	System global interrupt configuration register (SICFR)	R/W	0x0000_0000	8.5.1/8-8
0x04	System regular interrupt vector register (SIVCR)	R	0x0000_0000	8.5.2/8-9
0x08	System internal interrupt pending register (SIPNR_H)	R	0x0000_0000	8.5.3/8-11
0x0C	System internal interrupt pending register (SIPNR_L)	R	0x0000_0000	8.5.3/8-11
0x10	System internal interrupt group A priority register (SIPRR_A)	R/W	0x0530_9770	8.5.4/8-14
0x1C	System internal interrupt group D priority register (SIPRR_D)	R/W	0x0530_9770	8.5.5/8-14
0x20	System internal interrupt mask register (SIMSR_H)	R/W	0x0000_0000	8.5.6/8-15
0x24	System internal interrupt mask register (SIMSR_L)	R/W	0x0000_0000	8.5.6/8-15
0x28	System internal interrupt control register (SICNR)	R/W	0x0000_0000	8.5.7/8-17
0x2C	System external interrupt pending register (SEPNR)	R/W	Special	8.5.8/8-18
0x30	System mixed interrupt group A priority register (SMPRR_A)	R/W	0x0530_9770	8.5.8/8-18
0x34	System mixed interrupt group B priority register (SMPRR_B)	R/W	0x0530_9770	8.5.10/8-19
0x38	System external interrupt mask register (SEMSR)	R/W	0x0000_0000	8.5.11/8-20

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3C	System external interrupt control register (SECNR)	R/W	0x0000_0000	8.5.12/8-21
0x40	System error status register (SERSR)	R/W	0x0000_0000	8.5.13/8-22
0x44	System error mask register (SERMR)	R/W	—	8.5.14/8-23
0x48	System error control register (SERCR)	R/W	0x0000_0000	8.5.15/8-24
0x4C–0x4F	Reserved	—	—	—
0x50	System internal interrupt force register (SIFCR_H)	R/W	0x0000_0000	8.5.16/8-25
0x54	System internal interrupt force register (SIFCR_L)	R/W	0x0000_0000	8.5.16/8-25
0x58	System external interrupt force register (SEFCR)	R/W	0x0000_0000	8.5.17/8-26
0x5C	System error force register (SERFR)	R/W	0x0000_0000	8.5.18/8-26
0x60	System critical interrupt vector register (SCVCR)	R	0x0000_0000	8.5.19/8-27
0x64	System management interrupt vector register (SMVCR)	R	0x0000_0000	8.5.20/8-27
0x68–0xFF	Reserved	—	—	—
System Arbiter Registers				
0x00	Arbiter configuration register (ACR)	R/W	0x0000_0000/ 0x0010_0000[9]	6.2.1/6-2
0x04	Arbiter timers register (ATR)	R/W	0x00FF_00FF	6.2.2/6-4
0x0C	Arbiter event register (AER)	w1c	0x0000_0000	6.2.3/6-5
0x10	Arbiter interrupt definition register (AIDR)	R/W	0x0000_0000	6.2.4/6-6
0x14	Arbiter mask register (AMR)	R/W	0x0000_0000	6.2.5/6-7
0x18	Arbiter event attributes register (AEATR)	R	0x0000_0000[10]	6.2.6/6-7
0x1C	Arbiter event address register (AEADR)	R	0x0000_0000[10]	6.2.7/6-9
0x20	Arbiter event response register (AERR)	R/W	0x0000_0000	6.2.8/6-10
Reset Module				
0x0_0900	Reset configuration word low register (RCWLR)	R	0x0000_0000	4.5.1.1/4-30
0x0_0904	Reset configuration word high register (RCWHR)	R	0x0000_0000	4.5.1.2/4-30
0x0_0908	Reserved, should be cleared	—	—	—
0x0_090C	Reserved, should be cleared	—	—	—
0x0_0910	Reset status register (RSR)	R/W	0x0000_0000	4.5.1.3/4-31
0x0_0914	Reset mode register (RMR)	R/W	0x0000_0000	4.5.1.4/4-32
0x0_0918	Reset protection register (RPR)	R/W	0x0000_0000	4.5.1.5/4-33
0x0_091C	Reset control register (RCR)	R/W	0x0000_0000	4.5.1.6/4-33
0x0_0920	Reset control enable register (RCER)	R/W	0x0000_0000	4.5.1.7/4-34
0x0_0924–0x0_09FC	Reserved, should be cleared.	—	—	—
Clock Module				
0x0_0A00	System PLL mode register (SPMR)	R	0x*nnnn_nnnn*	4.5.2.1/4-35

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0A04	Output clock control register (OCCR)	R/W	0x0000_0000	4.5.2.2/4-36
0x0_0A08	System clock control register (SCCR)	R/W	0xFDFF_FFFF	4.5.2.3/4-37
0x0_0A0C–0x0_0AFC	Reserved, should be cleared	—	—	—
Power Management Control Module				
0x00B00	Power management controller configuration register (PMCCR)	R/W	0x0000_0000	5.8.2.1/5-60
0x00B04	Power management controller event register (PMCER)	R/W	0x0000_0000	5.8.2.2/5-61
0x00B08	Power management controller mask register (PMCMR)	R/W	0x0000_0000	5.8.2.3/5-62
0x00B0C–0x00BFC	Reserved	—	—	—
QUICC Engine Ports Interrupt Registers				
0x0_0C00–0x0_0C0B	Reserved	—	—	—
0x0_0C0C	CEPIER—QUICC Engine ports interrupt event register	R/W	Undefined	8.5.21/8-28
0x0_0C10	CEPIMR—QUICC Engine ports interrupt mask register	R/W	0x0000_0000	8.5.22/8-29
0x0_0C14	CEPICR—QUICC Engine ports interrupt control register	R/W	0x0000_0000	8.5.23/8-30
0x0_0C18–0x0_0CFF	Reserved	—	—	—
0x0_0D00–0x0_0FFF	Reserved	—	—	—
Clock Control DDR				
0x0_1000–0x0_100F	Reserved, should be cleared	—	—	—
0x0_1010	MCK enable register (MCKENR)	R/W	0xC000_0000	4.5.3/4-37
0x0_1014–0x0_10FF	Reserved, should be cleared	—	—	—
0x0_1100–0x0_1110	Reserved, should be cleared.	—	—	—
0x0_1114–0x0_13FF	Reserved, should be cleared	—	—	—
QUICC Engine Parallel I/O Ports Registers				
0x0_1400	CPODRA—QUICC Engine port A open drain register	R/W	0x0000_0000	3.4.3.1/3-15
0x0_1404	CPDATA—QUICC Engine port A data register	R/W	0x0000_0000	3.4.3.2/3-15
0x0_1408	CPDIR1A—QUICC Engine port A direction register 1	R/W	0x0000_0000	3.4.3.3/3-16
0x0_140C	CPDIR2A—QUICC Engine port A direction register 2	R/W	0x0000_0000	3.4.3.3/3-16
0x0_1410	CPPAR1A—QUICC Engine port A pin assignment register 1	R/W	0x0000_0000	3.4.3.4/3-17
0x0_1414	CPPAR2A—QUICC Engine port A pin assignment register 2	R/W	0x0000_0000	3.4.3.4/3-17
0x0_1418	CPODRB—QUICC Engine port B open drain register	R/W	0x0000_0000	3.4.3.1/3-15
0x0_141C	CPDATB—QUICC Engine port B data register	R/W	0x0000_0000	3.4.3.2/3-15

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_1420	CPDIR1B—QUICC Engine port B direction register 1	R/W	0x0000_0000	3.4.3.3/3-16
0x0_1424	CPDIR2B—QUICC Engine port B direction register 2	R/W	0x0000_0000	3.4.3.3/3-16
0x0_1428	CPPAR1B—QUICC Engine port B pin assignment register 1	R/W	0x0000_0000	3.4.3.4/3-17
0x0_142C	CPPAR2B—QUICC Engine port B pin assignment register 2	R/W	0x0000_0000	3.4.3.4/3-17
0x0_1430	CPODRC—QUICC Engine port C open drain register	R/W	0x0000_0000	3.4.3.1/3-15
0x0_1434	CPDATC—QUICC Engine port C data register	R/W	0x0000_0000	3.4.3.2/3-15
0x0_1438	CPDIR1C—QUICC Engine port C direction register 1	R/W	0x0000_0000	3.4.3.3/3-16
0x0_143C	CPDIR2C—QUICC Engine port C direction register 2	R/W	0x0000_0000	3.4.3.3/3-16
0x0_1440	CPPAR1C—QUICC Engine port C pin assignment register 1	R/W	0x0000_0000	3.4.3.4/3-17
0x0_1444	CPPAR2C—QUICC Engine port C pin assignment register 2	R/W	0x0000_0000	3.4.3.4/3-17
0x0_1448	CPODRD—QUICC Engine port D open drain register	R/W	0x0000_0000	3.4.3.1/3-15
0x0_144C	CPDATD—QUICC Engine port D data register	R/W	0x0000_0000	3.4.3.2/3-15
0x0_1450	CPDIR1D—QUICC Engine port D direction register 1	R/W	0x0000_0000	3.4.3.3/3-16
0x0_1454	CPDIR2D—QUICC Engine port D direction register 2	R/W	0x0000_0000	3.4.3.3/3-16
0x0_1458	CPPAR1D—QUICC Engine port D pin assignment register 1	R/W	0x0000_0000	3.4.3.4/3-17
0x0_145C	CPPAR2D—QUICC Engine port D pin assignment register 2	R/W	0x0000_0000	3.4.3.4/3-17
0x0_1460–0x0_1FFF	Reserved	—	—	—
DDR Memory Controller Memory Map				
0x0_2000	CS0_BNDS—Chip select memory bounds	R/W	0x0000_0000	9.4.1.1/9-9
0x0_2008–0x0_207F	Reserved	—	—	—
0x0_2080	CS0_CONFIG—Chip select configuration	R/W	0x0000_0000	9.4.1.2/9-10
0x0_2084–0x0_20FF	Reserved	—	—	—
0x0_2100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.3/9-11
0x0_2104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-12
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-14
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-16
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-18
0x0_2114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.8/9-21
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.9/9-22
0x0_211C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.10/9-23
0x0_2120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.11/9-24
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.12/9-26
0x0_2128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.13/9-26
0x0_2130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-27

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_2134–0x0_2144	Reserved	—	—	—
0x0_2148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	0x0000_0000	9.4.1.15/9-27
0x0_214C–0x0_2BF4	Reserved	—	—	—
0x0_2BF8	DDR_IP_REV1—DDR IP block revision 1	R	0x0002_0200	9.4.1.16/9-28
0x0_2BFC	DDR_IP_REV2—DDR IP block revision 2	R	0x0000_0000	9.4.1.17/9-28
0x0_2E00–0x0_2FFF	Reserved	—	—	—
I^2C Controller				
0x0_3000	I2CADR—I^2C address register	R/W	0x00	15.3.1.1/15-5
0x0_3004	I2CFDR—I^2C frequency divider register	R/W	0x00	15.3.1.2/15-5
0x0_3008	I2CCR—I^2C control register	R/W	0x00	15.3.1.3/15-6
0x0_300C	I2CSR—I^2C status register	R/W	0x81	15.3.1.4/15-7
0x0_3010	I2CDR—I^2C data register	R/W	0x00	15.3.1.5/15-9
0x0_3014	I2CDFSRR—I^2C digital filter sampling rate register	R/W	0x10	15.3.1.6/15-9
0x0_3018–0x0_30FF	Reserved, should be cleared	—	—	—
0x0_3100–0x0_3FFF	Reserved	—	—	—
DUART				
0x0_4000–0x0_44FF	Reserved, should be cleared	—	—	—
0x0_4500	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	16.3.1.1/16-5
0x0_4500	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	16.3.1.2/16-6
0x0_4500	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4501	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	16.3.1.4/16-8
0x0_4501	UDMB—ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4502	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	16.3.1.5/16-9
0x0_4502	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	16.3.1.6/16-10
0x0_4502	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	16.3.1.12/16-16
0x0_4503	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	16.3.1.7/16-11
0x0_4504	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	0x00	16.3.1.8/16-13
0x0_4505	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	16.3.1.9/16-14
0x0_4506	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	0x00	16.3.1.10/16-15
0x0_4507	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	16.3.1.11/16-16

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_4510	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	16.3.1.13/16-17
0x0_4600	URBR—ULCR[DLAB] = 0 UART2 receiver buffer register	R	0x00	16.3.1.1/16-5
0x0_4600	UTHR—ULCR[DLAB] = 0 UART2 transmitter holding register	W	0x00	16.3.1.2/16-6
0x0_4600	UDLB—ULCR[DLAB] = 1 UART2 divisor least significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4601	UIER—ULCR[DLAB] = 0 UART2 interrupt enable register	R/W	0x00	16.3.1.4/16-8
0x0_4601	UDMB_ULCR[DLAB] = 1 UART2 divisor most significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4602	UIIR—ULCR[DLAB] = 0 UART2 interrupt ID register	R	0x01	16.3.1.5/16-9
0x0_4602	UFCR—ULCR[DLAB] = 0 UART2 FIFO control register	W	0x00	16.3.1.6/16-10
0x0_4602	UAFR—ULCR[DLAB] = 1 UART2 alternate function register	R/W	0x00	16.3.1.12/16-16
0x0_4603	ULCR—ULCR[DLAB] = x UART2 line control register	R/W	0x00	16.3.1.7/16-11
0x0_4604	UMCR—ULCR[DLAB] = x UART2 MODEM control register	R/W	0x00	16.3.1.8/16-13
0x0_4605	ULSR—ULCR[DLAB] = x UART2 line status register	R	0x60	16.3.1.9/16-14
0x0_4606	UMSR—ULCR[DLAB] = x UART2 MODEM status register	R	0x00	16.3.1.10/16-15
0x0_4607	USCR—ULCR[DLAB] = x UART2 scratch register	R/W	0x00	16.3.1.11/16-16
0x0_4610	UDSR—ULCR[DLAB] = x UART2 DMA status register	R	0x01	16.3.1.13/16-17
0x0_4700–0x0_4FFF	Reserved, should be cleared	—	—	—
Local Bus Controller (LBC) Registers				
0x0_5000	BR0—Base register 0	R/W	0x0000_RR01[11]	10.3.1.1/10-8
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020–0x0_5038	Reserved	—	—	—
0x0_5004	OR0—Option register 0	R/W	0x0000_0FF7	10.3.1.2/10-10
0x0_500C	OR1—Option register 1		0x0000_0000	
0x0_5014	OR2—Option register 2			
0x0_501C	OR3—Option register 3			
0x0_5024–0x0_503C	Reserved	—	—	—
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	10.3.1.3/10-14
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	10.3.1.4/10-15
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	10.3.1.4/10-15
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	10.3.1.4/10-15
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	10.3.1.5/10-17

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	10.3.1.6/10-18
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	10.3.1.7/10-18
0x0_50B0	LTESR—Transfer error status register	Read/bit-reset	0x0000_0000	10.3.1.8/10-19
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	10.3.1.9/10-20
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	10.3.1.10/10-21
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	10.3.1.11/10-22
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	10.3.1.12/10-23
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	10.3.1.13/10-23
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	10.3.1.14/10-24
DMA Registers				
0x0_8030	OMISR—Outbound message interrupt status register	Mixed	0x0000_0000	12.3.1/12-3
0x0_8034	OMIMR—Outbound message interrupt mask register	R/W	0x0000_0000	12.3.2/12-5
0x0_8050	IMR0—Inbound message register 0	R/W	0x0000_0000	12.3.3/12-6
0x0_8054	IMR1—Inbound message register 1	R/W	0x0000_0000	12.3.3/12-6
0x0_8058	OMR0—Outbound message register 0	R/W	0x0000_0000	12.3.4/12-6
0x0_805C	OMR1—Outbound message register 1	R/W	0x0000_0000	12.3.4/12-6
0x0_8060	ODR—Outbound doorbell register	R/W	0x0000_0000	12.3.5/12-6
0x0_8068	IDR—Inbound doorbell register	R/W	0x0000_0000	12.3.5/12-6
0x0_8080	IMISR—Inbound message interrupt status register	Mixed	0x0000_0000	12.3.6/12-8
0x0_8084	IMIMR—Inbound message interrupt mask register	R/W	0x0000_0000	12.3.7/12-9
0x0_8100	DMAMR0—DMA 0 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8104	DMASR0—DMA 0 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8108	DMACDAR0—DMA 0 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8110	DMASAR0—DMA 0 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8118	DMADAR0—DMA 0 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_8120	DMABCR0—DMA 0 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_8124	DMANDAR0—DMA 0 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_8180	DMAMR1—DMA 1 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8184	DMASR1—DMA 1 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8188	DMACDAR1—DMA 1 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8190	DMASAR1—DMA 1 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8198	DMADAR1—DMA 1 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_81A0	DMABCR1—DMA 1 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_81A4	DMANDAR1—DMA 1 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_8200	DMAMR2—DMA 2 mode register	R/W	0x0000_0000	12.3.8.1/12-10

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8204	DMASR2—DMA 2 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8208	DMACDAR2—DMA 2 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8210	DMASAR2—DMA 2 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8218	DMADAR2—DMA 2 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_8220	DMABCR2—DMA 2 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_8224	DMANDAR2—DMA 2 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_8280	DMAMR3—DMA 3 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8284	DMASR3—DMA 3 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8288	DMACDAR3—DMA 3 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8290	DMASAR3—DMA 3 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8298	DMADAR3—DMA 3 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_82A0	DMABCR3—DMA 3 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_82A4	DMANDAR3—DMA 3 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_82A8	DMAGSR—DMA general status register	R	0x0000_0000	12.3.8.8/12-16
0x0_82B0–0x0_82FF	Reserved	—	—	—
PCI Configuration Access Registers				
0x0	PCI_CONFIG_ADDRESS	W	0x0000_0000	13.3.1.1/13-13
0x4	PCI_CONFIG_DATA	R/W	0x0000_0000	13.3.1.2/13-14
0x8	PCI_INT_ACK	R	—	13.3.1.3/13-15
Sequencer (IOS)				
0x00	POTAR0—PCI outbound translation address register 0	R/W	0x0000_0000	11.4.1/11-3
0x08	POBAR0—PCI outbound base address register 0	R/W	0x0000_0000	11.4.2/11-3
0x10	POCMR0—PCI outbound comparison mask register 0	R/W	0x0000_0000	11.4.3/11-4
0x18	POTAR1—PCI outbound translation address register 1	R/W	0x0000_0000	11.4.1/11-3
0x20	POBAR1—PCI outbound base address register 1	R/W	0x0000_0000	11.4.2/11-3
0x28	POCMR1—PCI outbound comparison mask register 1	R/W	0x0000_0000	11.4.3/11-4
0x30	POTAR2—PCI outbound translation address register 2	R/W	0x0000_0000	11.4.1/11-3
0x38	POBAR2—PCI outbound base address register 2	R/W	0x0000_0000	11.4.2/11-3
0x40	POCMR2—PCI outbound comparison mask register 2	R/W	0x0000_0000	11.4.3/11-4
0x48	POTAR3—PCI outbound translation address register 3	R/W	0x0000_0000	11.4.1/11-3
0x50	POBAR3—PCI outbound base address register 3	R/W	0x0000_0000	11.4.2/11-3
0x58	POCMR3—PCI outbound comparison mask register 3	R/W	0x0000_0000	11.4.3/11-4
0x60	POTAR4—PCI outbound translation address register 4	R/W	0x0000_0000	11.4.1/11-3
0x68	POBAR4—PCI outbound base address register 4	R/W	0x0000_0000	11.4.2/11-3
0x70	POCMR4—PCI outbound comparison mask register 4	R/W	0x0000_0000	11.4.3/11-4

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x78	POTAR5—PCI outbound translation address register 5	R/W	0x0000_0000	11.4.1/11-3
0x80	POBAR5—PCI outbound base address register 5	R/W	0x0000_0000	11.4.2/11-3
0x88	POCMR5—PCI outbound comparison mask register 5	R/W	0x0000_0000	11.4.3/11-4
0xF0	PMCR—Power management control register	R/W	0x0000_0000	11.4.4/11-5
0xF8	DTCR—Discard timer control register	R/W	0x0000_0000	11.4.5/11-6
Security Engine Address Map Registers				
Controller Registers				
0x3_0000–0x3_0FFF	Reserved, should be cleared	—	—	—
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000_0000_0000	14.6.4.2/14-71
0x3_1010	ISR—Interrupt status register	R	0x0000_0000_0000_0000	14.6.4.3/14-73
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000_0000_0000	14.6.4.4/14-74
0x3_1020	ID—Identification register	R	0x0000_0000_0000_00A0	14.6.4.5/14-75
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	14.6.4.1/14-70
0x3_1030	MCR—Master control register	R/W	0000_0000_0000_0000	14.6.4.7/14-76
Channel				
0x3_1108	CCCR—Crypto-channel configuration register	R/W	0x0000_0000_0000_0000	14.5.1.1/14-57
0x3_1110	CCPSR—Crypto-channel pointer status register	R	0x0000_0000_0000_0007	14.5.1.2/14-59
0x3_1140	CDPR—Crypto-channel current descriptor pointer register	R	0x0000_0000_0000_0000	14.5.1.3/14-64
0x3_1148	FF—Crypto-channel fetch FIFO address register	W	0x0000_0000_0000_0000	14.5.1.4/14-64
0x3_1180–0x3_11BF	DB*n*—Crypto-channel descriptor buffers [0–7]	R	0x0000_0000_0000_0000	14.5.1.5/14-65
0x3_1BF8	IP block revision register	R	0x0000_0000_0000_00A0	14.5.2.1/14-66
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000_0000_0000	14.4.1.1/14-20
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000_0000_0000	14.4.1.2/14-21

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000 _0000_0000	14.4.1.3/14-22
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000 _0000_0000	14.4.1.4/14-23
0x3_2028	DEUSR—DEU status register	R	0x0000_0000 _0000_0000	14.4.1.5/14-24
0x3_2030	DEUISR—DEU interrupt status register	R	0x0000_0000 _0000_0000	14.4.1.6/14-25
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000 _0000_3000	14.4.1.7/14-26
0x3_2050	DEUEMR—DEU end-of-message register	W	0x0000_0000 _0000_0000	14.4.1.8/14-28
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000 _0000_0000	14.4.1.9/14-28
0x3_2400	DEUK1—DEU key 1 register	W	—	14.4.1.10/14-29
0x3_2408	DEUK2—DEU key 2 register	W	—	14.4.1.10/14-29
0x3_2410	DEUK3—DEU key 3 register	W	—	14.4.1.10/14-29
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000 _0000_0000	14.4.1.11/14-29
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000 _0000_0000	14.4.3.1/14-41
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000 _0000_0000	14.4.3.2/14-44
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000 _0000_0000	14.4.3.3/14-45
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000 _0000_0000	14.4.3.4/14-45
0x3_4028	AESUSR—AESU status register	R	0x0000_0000 _0000_0000	14.4.3.5/14-46
0x3_4030	AESUISR—AESU interrupt status register	R	0x0000_0000 _0000_0000	14.4.3.6/14-47
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000 _0000_1000	14.4.3.7/14-49
0x3_4050	AESUEMR—AESU end-of-message register	W	0x0000_0000 _0000_0000	14.4.3.8/14-50
0x3_4100	AESU context memory registers	R/W	0x0000_0000 _0000_0000	14.4.3.9/14-51
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000 _0000_0000	14.4.3.9/14-51
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000 _0000_0000	14.4.3.9/14-51

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000 _0000_0000	14.4.2.1/14-29
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000 _0000_0000	14.4.2.3/14-33
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000 _0000_0000	14.4.2.4/14-33
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000 _0000_0000	14.4.2.5/14-34
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000 _0000_0000	14.4.2.6/14-35
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000 _0000_0000	14.4.2.7/14-36
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000 _0000_1000	14.4.2.8/14-37
0x3_6040	MDEU ICV size register	R/W	0x0000_0000_ 0000_0000	14.4.2.9/14-38
0x3_6050	MDEUEMR—MDEU end-of-message register	W	0x0000_0000 _0000_0000	14.4.2.10/14-39
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000 _0000_0000	14.4.2.11/14-39
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000 _0000_0000	14.4.2.12/14-40
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000 _0000_0000	14.4.2.13/14-41
QUICC Engine 1.0 Block				
0x10_0000 – 0x1F_FFFF	See Table 2-3 and Table 2-4 for more information.	—	—	—

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

[10] The registers AEATR and AEADR are affected only by the assertion of $\overline{\text{PORESET}}$

[11] Port size for BR0 is configured from the value of RCWH[ROMLOC], which is loaded during reset, hence ‘RR’ is either 0x08 or 0x10.

2.4 QUICC Engine Internal Memory Map

The QUICC Engine block’s internal memory resources are mapped within a contiguous block of memory. The size of the internal space is 1 Mbyte. The location of the QUICC Engine internal memory space within the global system memory space can be mapped (aligned to a 1-Mbyte boundary) through an implementation-specific special register, see Section 5.2.4.1, “Internal Memory Map Registers Base Address Register (IMMRBAR),” for more information. Note that the last 768 Kbytes of the QUICC Engine internal memory space are reserved for future expansions. Any access to reserved regions will result in undefined behavior.

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

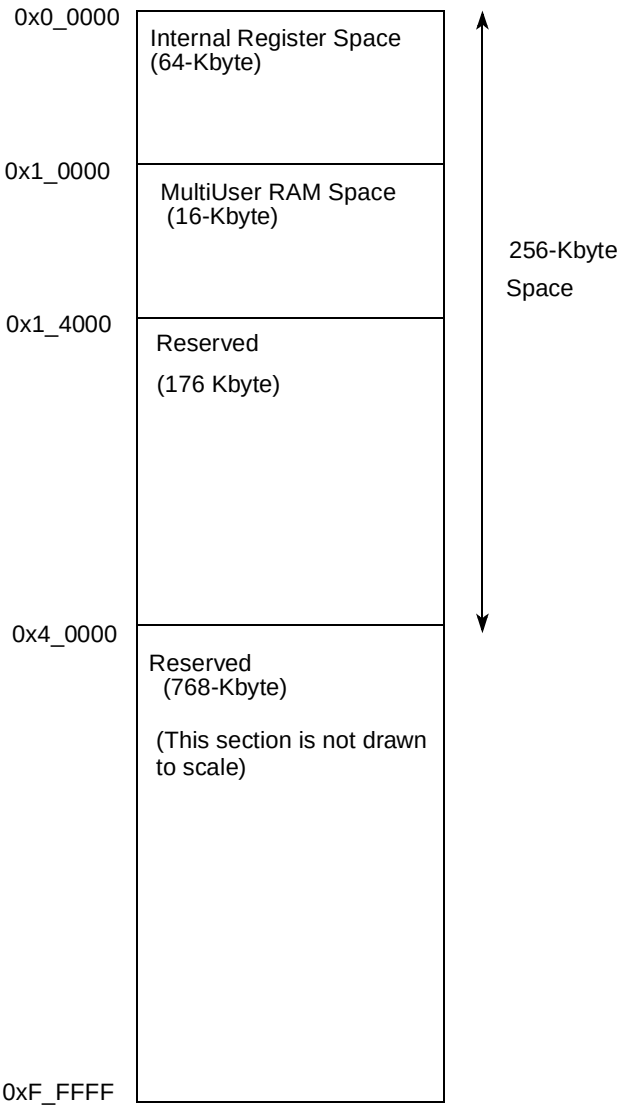


Figure 2-1. QUICC Engine 1.0 High-Level Memory Map

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

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

Internal Address	Abbreviation	Name	Size	Comments
0x0_0000–0x0_FFFF	Register Space			
0x0_0000–0x0_3FFF	General Space			
0x0_0000–0x0_003F	I-RAM	Instruction RAM registers	64 bytes	—
0x0_0040–0x0_007F	Reserved	—	64 bytes	—
0x0_0080–0x0_00FF	IRQ	Interrupt controller	128 bytes	—
0x0_0100–0x0_01FF	RISC Config	RISC configuration register	256 bytes	—
0x0_0200–0x0_03FF	Reserved	—	512 bytes	—
0x0_0400–0x0_043F	QUICC Engine Mux	QUICC Engine clock multiplexer registers	64 bytes	—
0x0_0440–0x0_047F	Timers	QUICC Engine timers	64 bytes	—
0x0_0480–0x0_04BF	Reserved	—	64 bytes	—
0x0_04C0–0x0_04FF	SPI1	SPI1 registers	64 bytes	—
0x0_0500–0x0_053F	SPI2	SPI2 registers	64 bytes	—
0x0_0540–0x0_063F	Reserved	—	256 bytes	—
0x0_0640–0x0_067F	BRG	Baud rate generator registers	64 bytes	—
0x0_0680–0x0_06FF	Reserved	—	128 bytes	—
0x0_06C0–0x0_06FF	USB1.0	USB 1.0 registers	64 bytes	—
0x0_0700–0x0_077F	SI1	SI1 registers	128 bytes	—
0x0_0780–0x0_07FF	Reserved	—	128 bytes	—
0x0_0800–0x0_0FFF	Reserved	—	2K bytes	—
0x0_1000–0x0_17FF	SI1RT	SI1 routing table	2K bytes	—
0x0_1800–0x0_1FFF	Reserved	—	2K bytes	—
0x0_2000–0x0_21FF	UCC1	UCC1 registers	512 bytes	—
0x0_2200–0x0_23FF	UCC3	UCC3 registers	512 bytes	—
0x0_2400–0x0_25FF	UCC5	UCC5 registers	512 bytes	—
0x0_2600–0x0_2DFF	Reserved	—	2048 bytes	—
0x0_2E00–0x0_2FFF	UPC1	Utopia POS controller 1	512 bytes	—
0x0_3000–0x0_31FF	UCC2	UCC2 registers	512 bytes	—
0x0_3200–0x0_33FF	UCC4	UCC4 registers	512 bytes	—
0x0_3400–0x0_3FFF	Reserved	—	2048 bytes	—
0x0_4000–0x0_407F	SDMA	Serial DMA	128 bytes	—

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

Internal Address	Abbreviation	Name	Size	Comments
0x0_4080–0x0_7FFF	Debug Space			
0x0_4080–0x0_42FF	Reserved	—	256 bytes	—
0x0_4300–0x0_43FF	Reserved	—	256 bytes	—
0x0_4400–0x0_45FF	Reserved	—	256 bytes	—
0x0_4600–0x0_467F	RISC1 TRBR	RISC1 trace buffer registers	128 bytes	—
0x0_4680–0x0_46FF	Reserved	—	128 bytes	—
0x0_4700–0x0_477F	Reserved	—	128 bytes	—
0x0_4780–0x0_47FF	Reserved	—	128 bytes	—
0x0_4800–0x0_7FFF	Reserved	—	14 Kbytes	—
0x0_5000–0x0_7FFF	Reserved	—	12 Kbytes	—
0x0_8000–0x3_FFFF	RAM Space			
0x0_8000–0x0_FFFF	Reserved	—	32 Kbytes	—
0x1_0000–0x1_3FFF	MURAM	Multi-user RAM	16 Kbytes	—
0x1_4000–0x1_BFFF	Reserved	—	32 Kbytes	—
0x1_C000–0x3_FFFF	Reserved	—	144 Kbytes	
0x4_0000–0xF_FFFF	Reserved	—	768 Kbytes	

Table 2-4 shows the detailed QUICC Engine memory map. The Internal Address field is the hexidecimal offset from the QUICC Engine base address allocated in the system. All registers are reset by ‘soft reset’ condition except for the ones marked in the ‘comments’ column of the table.

Table 2-4. Detailed QUICC Engine Memory Map

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
Internal Register Space					
I-RAM Registers					
0x0_0000	IADD	I-RAM address register	4 bytes	19.3.5/19-11	—
0x0_0004	IDATA	I-RAM data register	4 bytes	19.3.6/19-12	—
0x0_0008–0x0_007F	Reserved	—	120 bytes	—	—
Interrupt Controller					
0x0_0080	CICR	QUICC Engine system interrupt configuration register	4 bytes	18.3.1/18-24	—
0x0_0084	CIVEC	QUICC Engine system interrupt vector register	4 bytes	18.3.12/18-34	—
0x0_0088	CRIPNR	QUICC Engine RISC interrupt pending register	4 bytes	18.3.12/18-34	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_008C	CIPNR	QUICC Engine system interrupt pending register	4 bytes	18.3.10/18-33	—
0x0_0090	CIPXCC	QUICC Engine interrupt priority register	4 bytes	18.3.3/18-27	—
0x0_0094	CIPYCC	QUICC Engine interrupt priority register	4 bytes	18.3.5/18-29	—
0x0_0098	CIPWCC	QUICC Engine interrupt priority register	4 bytes	18.3.6/18-29	—
0x0_009C	CIPZCC	QUICC Engine interrupt priority register	4 bytes	18.3.7/18-30	—
0x0_00A0	CIMR	QUICC Engine system interrupt mask register	4 bytes	18.3.11/18-33	—
0x0_00A4	CRIMR	QUICC Engine RISC interrupt mask register	4 bytes	18.3.11/18-33	—
0x0_00A8	CICNR	QUICC Engine system interrupt control register	4 bytes	18.3.2/18-25	—
0x0_00B0	CIPRTA	QUICC Engine system interrupt priority register for RISC tasks A	4 bytes	18.3.8/18-31	—
0x0_00B4	CIPRTB	QUICC Engine system interrupt priority register for RISC tasks B	4 bytes	18.3.9/18-32	—
0x0_00B8–0x0_00BB	Reserved	—	4 bytes	—	—
0x0_00BC	CRICR	QUICC Engine system RISC interrupt control register	4 bytes	18.3.3/18-27	—
0x0_00C0–0x0_00DC	Reserved	—	32 bytes	—	—
0x0_00E0	CHIVEC	QUICC Engine high system interrupt vector register	4 bytes	18.3.15/18-37	—
0x0_00E4–0x0_00FF	Reserved	—	28 bytes	—	—
Communications Processor					
0x0_0100	CECR	QUICC Engine command register	4 bytes	19.3.1/19-2	Hard reset
0x0_0104	CECCR	QUICC Engine controller configuration register	4 bytes	19.3.8/19-13	Hard reset
0x0_0108	CECDR	QUICC Engine command data register	4 bytes	19.3.2/19-9	—
0x0_010C–0x0_0115	Reserved	—	10 bytes	—	—
0x0_0116	CETER	QUICC Engine timer event register	2 bytes	19.4.4/19-19	—
0x0_0118	Reserved	—	2 bytes	—	—
0x0_011A	CETMR	QUICC Engine timers mask register	2 bytes	19.4.4/19-19	—
0x0_011C	CETSCR	QUICC Engine time-stamp timer control register	4 bytes	19.3.9/19-14	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_0120	CETSR1	QUICC Engine time-stamp register 1	4 bytes	19.3.9/19-14	—
0x0_0124	CETSR2	QUICC Engine time-stamp register 2	4 bytes	19.3.9/19-14	—
0x0_0128–0x0_012F	Reserved	—	8 bytes	—	—
0x0_0130	CEVTER	QUICC Engine virtual tasks event register	4 bytes	19.3.3/19-9	—
0x0_0134	CEVTMR	QUICC Engine virtual tasks mask register	4 bytes	19.3.3/19-9	—
0x0_0138	CERCR	QUICC Engine RAM control register	2 bytes	19.3.4/19-10	—
0x0_013C–0x0_015F	Reserved	—	36 bytes	—	—
0x0_0160	CEEXE1	QUICC Engine external request 1 event register	2 bytes	19.5.1/19-20	—
0x0_0162–0x0_0163	Reserved	—	2 bytes	—	—
0x0_0164	CEEXM1	QUICC Engine external request 1 mask register	2 bytes	19.5.1/19-20	—
0x0_0166–0x0_0167	Reserved	—	2 bytes	—	—
0x0_0168	CEEXE2	QUICC Engine external request 2 event register	2 bytes	19.5.1/19-20	—
0x0_016A–0x0_016B	Reserved	—	2 bytes	—	—
0x0_016C	CEEXM2	QUICC Engine external request 2 mask register	2 bytes	19.5.1/19-20	—
0x0_016E–0x0_016F	Reserved	—	2 bytes	—	—
0x0_0170	CEEXE3	QUICC Engine external request 3 event register	2 bytes	19.5.1/19-20	—
0x0_0172–0x0_0173	Reserved	—	2 bytes	—	—
0x0_0174	CEEXM3	QUICC Engine external request 3 mask register	2 bytes	19.5.1/19-20	—
0x0_0176–0x0_0177	Reserved	—	2 bytes	—	—
0x0_0178	CEEXE4	QUICC Engine external request 4 event register	2 bytes	19.5.1/19-20	—
0x0_017A–0x0_017B	Reserved	—	2 bytes	—	—
0x0_017C	CEEXM4	QUICC Engine external request 4 mask register	2 bytes	19.5.1/19-20	—
0x0_017E–0x0_017F	Reserved	—	2 bytes	—	—
0x0_01A0–0x0_01C0	Reserved	—	20 bytes	—	—
0x0_01BC–0x0_01FF	Reserved	Future	192 bytes	—	—
0x0_0200–0x0_03FF	Reserved	—	64 bytes	—	—
QUICC Engine Multiplexer					

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_0400	CMXGCR	CMX general clock route register	4 bytes	20.5.1/20-8	Hard reset
0x0_0404	CMXSI1CR_L	CMX SI1 clock route low register	4 bytes	20.5.2/20-10	Hard reset
0x0_040C	CMXSI1SYR	CMX SI1 SYNC route register	4 bytes	20.5.3/20-12	Hard reset
0x0_0410	CMXUCR1	CMX UCC1, UCC3 clock route register	4 bytes	20.5.4/20-14	Hard reset
0x0_0414	CMXUCR2	CMX UCC5 clock route register	4 bytes	20.5.5/20-17	Hard reset
0x0_0418	CMXUCR3	CMX UCC2, UCC4 clock route register	4 bytes	20.5.6/20-18	Hard reset
0x0_041C	Reserved	—	4 bytes	—	Hard reset
0x0_0420	CMXUPCR	CMX UPC clock route register	4 bytes	20.5.7/20-21	Hard reset
0x0_0424	Reserved	—	4 bytes	—	Hard reset
0x0_0428–0x0_043F	Reserved	—	24 bytes	—	Hard reset
QUICC Engine Timers					
0x0_0440	GTCFR1	Timer 1 and Timer 2 global configuration register	1 byte	20.10.4.1/20-31	—
0x0_0441	Reserved	—	3 bytes	—	—
0x0_0444	GTCFR2	Timer 3 and timer 4 global configuration register	1 byte	20.10.4.1/20-31	—
0x0_0445–0x0_044F	Reserved	—	11 bytes	—	—
0x0_0450	GTMDR1	Timer 1 mode register	2 bytes	20.10.4.2/20-34	—
0x0_0452	GTMDR2	Timer 2 mode register	2 bytes		—
0x0_0454	GTRFR1	Timer 1 reference register	2 bytes	20.10.4.3/20-35	—
0x0_0456	GTRFR2	Timer 2 reference register	2 bytes		—
0x0_0458	GTCPR1	Timer 1 capture register	2 bytes	20.10.4.4/20-35	—
0x0_045A	GTCPR2	Timer 2 capture register	2 bytes		—
0x0_045C	GTCNR1	Timer 1 counter	2 bytes	20.10.4.5/20-36	—
0x0_045E	GTCNR2	Timer 2 counter	2 bytes		—
0x0_0460	GTMDR3	Timer 3 mode register	2 bytes	20.10.4.2/20-34	—
0x0_0462	GTMDR4	Timer 4 mode register	2 bytes		—
0x0_0464	GTRFR3	Timer 3 reference register	2 bytes	20.10.4.3/20-35	—
0x0_0466	GTRFR4	Timer 4 reference register	2 bytes		—
0x0_0468	GTCPR3	Timer 3 capture register	2 bytes	20.10.4.4/20-35	—
0x0_046A	GTCPR4	Timer 4 capture register	2 bytes		—
0x0_046C	GTCNR3	Timer 3 counter	2 bytes	20.10.4.5/20-36	—
0x0_046E	GTCNR4	Timer 4 counter	2 bytes		—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_0470	GTEVR1	Timer 1 event register	2 bytes	20.10.4.6/20-36	—
0x0_0472	GTEVR2	Timer 2 event register	2 bytes		—
0x0_0474	GTEVR3	Timer 3 event register	2 bytes		—
0x0_0476	GTEVR4	Timer 4 event register	2 bytes		—
0x0_0478	GTPS1	Timer 1 prescale register	2 bytes	20.10.4.7/20-37	—
0x0_047A	GTPS2	Timer 2 prescaler register	2 bytes		—
0x0_047C	GTPS3	Timer 3 prescaler register	2 bytes		—
0x0_047E	GTPS4	Timer 4 prescaler register	2 bytes		—
0x0_0480–0x0_04BF	Reserved	—	64 bytes	—	—
SPI SPI1_BASE = 0x004C0, SPI2_BASE=0x00500					
SPIn_BASE–SPIn_BASE+0x1F	Reserved	—	16 bytes	—	—
SPIn_BASE+0x20	SPMODE	SPI mode register	4 bytes	21.3.1/21-10 21.9.2/21-26	—
SPIn_BASE+0x24	Reserved	—	2 bytes	—	—
SPIn_BASE+0x26	SPIE	SPI event register	1 byte	21.3.2/21-13 21.9.3/21-27	—
SPIn_BASE+0x27	Reserved	—	3 bytes	—	—
SPIn_BASE+0x2A	SPIM	SPI mask register	1 byte	21.3.2/21-13 21.9.4/21-29	—
SPIn_BASE+0x2B	Reserved	—	2 bytes	—	—
SPIn_BASE+0x2D	SPCOM	SPI command register	1 byte	21.3.3/21-15 21.9.5/21-30	—
SPIn_BASE+0x2E	Reserved	—	2 bytes	—	—
SPIn_BASE+0x30	SPITD	SPI transmit data register (CPU mode)	4 bytes	21.9.6/21-30	—
SPIn_BASE+0x34	SPIRD	SPI receive data register (CPU mode)	4 bytes	21.9.7/21-31	—
SPIn_BASE+0x38–SPIn_BASE+0x3F	Reserved	—	8 bytes	—	—
0x0_0540–0x0_063F	Reserved	—	256 bytes	—	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
BRG BRG_BASE = 0x00640					
BRG_BASE+0x00	BRGC1	BRG1 configuration register	4 bytes	20.7/20-23	—
BRG_BASE+0x04	BRGC2	BRG2 configuration register	4 bytes		
BRG_BASE+0x08	BRGC3	BRG3 configuration register	4 bytes		
BRG_BASE+0x0C	BRGC4	BRG4 configuration register	4 bytes		
BRG_BASE+0x10	BRGC5	BRG5 configuration register	4 bytes		
BRG_BASE+0x14	BRGC6	BRG6 configuration register	4 bytes		
BRG_BASE+0x18	BRGC7	BRG7 configuration register	4 bytes		
BRG_BASE+0x1C	BRGC8	BRG8 configuration register	4 bytes		
BRG_BASE+0x20	BRGC9	BRG9 configuration register	4 bytes		
BRG_BASE+0x24	BRGC10	BRG10 configuration register	4 bytes		
BRG_BASE+0x28	BRGC11	BRG11 configuration register	4 bytes		
BRG_BASE+0x2C– BRG_BASE+0x34	Reserved	—	12 bytes	—	—
BRG_BASE+0x38	BRGC15	BRG15 configuration register	4 bytes	20.7/20-23	—
BRG_BASE+0x3C	BRGC16	BRG16 configuration register	4 bytes		
BRG_BASE+0x40– BRG_BASE+0x7F	Reserved	—	64 bytes	—	—
USB 1.0					
0x0_06C0	USMOD	USB mode register	1 byte	36.4.7.1/36-16	—
0x0_06C1	USADD	USB address register	1 byte	36.4.7.2/36-17	—
0x0_06C2	USCOM	USB command register	1 byte	36.4.7.4/36-19	—
0x0_06C3	Reserved	—	1 byte	—	—
0x0_06C4	USEP0	USB endpoint register 0	2 bytes	36.4.7.3/36-18	—
0x0_06C6	USEP1	USB endpoint register 1	2 bytes	36.4.7.3/36-18	—
0x0_06C8	USEP2	USB endpoint register 2	2 bytes	36.4.7.3/36-18	—
0x0_06CA	USEP3	USB endpoint register 3	2 bytes	36.4.7.3/36-18	—
0x0_06CC–0x0_06CF	Reserved	—	4 bytes	—	—
0x0_06D0	USBER	USB event register	2 bytes	36.4.7.5/36-20	—
0x0_06D2–0x0_06D3	Reserved	—	2 bytes	—	—
0x0_06D4	USBMR	USB mask register	2 bytes	36.4.7.6/36-21	—
0x0_06D6	Reserved	—	1 byte	—	—
0x0_06D7	USBS	USB status register	1 byte	36.4.7.7/36-21	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_06D8	USSFT	USB start of frame timer	4 bytes	36.4.7.8/36-22	—
0x0_06DC–0x0_06DF	Reserved	—	4 bytes	—	—
0x0_06E0–0x0_06FF	Reserved	—	32 bytes	—	—
SI1 Registers SI1_BASE=0x00700					
SI1_BASE+0x00	SIAMR1	SI1 TDMA mode register	2 bytes	32.6.3/32-13	—
SI1_BASE+0x02	SIBMR1	SI1 TDMB mode register	2 bytes	32.6.3/32-13	—
SI1_BASE+0x04	SICMR1	SI1 TDMC mode register	2 bytes	32.6.3/32-13	—
SI1_BASE+0x06	SIDMR1	SI1 TDMD mode register	2 bytes	32.6.3/32-13	—
SI1_BASE+0x08	SIGLMR1_H	SI1 global mode register high	1 byte	32.6.2/32-13	—
SI1_BASE+0x09	Reserved	Must be cleared	1 byte	—	—
SI1_BASE+0x0A	SICMDR1_H	SI1 command register high	1 byte	32.6.5/32-21	—
SI1_BASE+0x0B	Reserved	Must be cleared	1 byte	—	—
SI1_BASE+0x0C	SISTR1_H	SI1 status register high	1 byte	32.6.6/32-22	—
SI1_BASE+0x0D	Reserved	Must be cleared	1 byte	—	—
SI1_BASE+0x0E	SIRSR1_H	SI1 RAM shadow address register high	2 bytes	32.6.4/32-20	—
SI1_BASE+0x10	SITARC1	SI1 RAM counter Tx TDMA	1 byte	32.6.7/32-23	—
SI1_BASE+0x11	SITBRC1	SI1 RAM counter Tx TDMB	1 byte		
SI1_BASE+0x12	SITCRC1	SI1 RAM counter Tx TDMC	1 byte		
SI1_BASE+0x13	SITDRC1	SI1 RAM counter Tx TDMD	1 byte		
SI1_BASE+0x14	SIRARC1	SI1 RAM counter Rx TDMA	1 byte	32.6.7/32-23	—
SI1_BASE+0x15	SIRBRC1	SI1 RAM counter Rx TDMB	1 byte		
SI1_BASE+0x16	SIRCRC1	SI1 RAM counter Rx TDMC	1 byte		
SI1_BASE+0x17	SIRDRC1	SI1 RAM counter Rx TDMD	1 byte		
SI1_BASE+0x18–0x40	Reserved	—	44 bytes	—	—
SI1_BASE+0x44	SIENS1	SI1 extended diagnostic mode register	1 byte	32.6.8/32-23	—
SI1_BASE+0x45–SI1_BASE+0x7F	Reserved	—	—	—	—
SI1_BASE+0x80–SI1_BASE+0xFF	Reserved	—	—	—	—
0x0_0800–0x0_0FFF	Reserved	—	2 Kbytes	—	—
SI Routing Tables					
0x0_1000–0x0_13FF	SITxRAM	SI1 Tx routing table	1 Kbyte	32.7.1/32-23	—
0x0_1400–0x0_17FF	SIRxRAM	SI1 Rx routing table	1 Kbyte	32.7.1/32-23	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_1800–0x0_1FFF	Reserved	—	2 Kbytes	—	—
UCCx: x=1,2,3,4,5					
UCC1_BASE=0x0_2000; UCC3_BASE=0x0_2200; UCC5_BASE=0x0_2400; UCC2_BASE=0x0_3000; UCC4_BASE=0x0_3200;					
Slow Mode (UART, BISYNC, QMC)					
UCC*x*_BASE+0x0	GUMR_Lx	UCCx general mode register (low)	4 bytes	26.4.1/26-4	—
UCC*x*_BASE+0x4	GUMR_Hx	UCCx general mode register (high)	4 bytes	26.4.1/26-4	—
UCC*x*_BASE+0x8	UPSMRx	UCCx protocol-specific mode register	2 bytes	24.3.4/24-9 25.3.2.5/25-9	—
UCC*x*_BASE+0xA	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0xC	UTODRx	UCCx transmit on demand register	2 bytes	22.3.4/22-6	—
UCC*x*_BASE+0xE	UDSRx	UCCx data synchronization register	2 bytes	—	—
UCC*x*_BASE+0x10	UCCEx	UCCx event register	2 bytes	24.3.7/24-15 25.3.2.8/25-14 34.5.2/34-29	—
UCC*x*_BASE+0x12	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0x14	UCCMx	UCCx mask register	2 bytes	24.3.7/24-15 25.3.2.8/25-14 34.5.2/34-29	—
UCC*x*_BASE+0x16	Reserved	—	1 byte	—	—
UCC*x*_BASE+0x17	UCCSx	UCCx status register	1 byte	24.3.8/24-18	—
UCC*x*_BASE+0x18 –UCC*x*_BASE+0x1FF	Reserved	—	—	—	—
Fast Mode (Ethernet, ATM, HDLC, Transparent)					
UCC*x*_BASE+0x0	GUMRx	UCCx general mode register	4 bytes	26.4.2.1/26-6	—
UCC*x*_BASE+0x4	UPSMRx	UCCx protocol-specific mode register	4 bytes	27.2.2.2/27-7 28.4.2.2/28-6	—
UCC*x*_BASE+0x8	UTODRx	UCCx transmit on demand register	2 bytes	22.3.4/22-6	—
UCC*x*_BASE+0xA	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0xC	UDSRx	UCCx data synchronization register	2 bytes	26.4.5/26-11	—
UCC*x*_BASE+0xE	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0x10	UCCEx	UCCx event register	4 bytes	27.2.2.5/27-12 28.4.2.5/28-10	—
UCC*x*_BASE+0x14	UCCMx	UCCx mask register	4 bytes	27.2.2.5/27-12 28.4.2.5/28-10	—
UCC*x*_BASE+0x18	UCCSx	UCCx status register	1 byte	27.2.2.6/27-15	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UCC*x*_BASE+0x19 –UCC*x*_BASE+0x1F	Reserved	—	7 bytes	—	—
UCC*x*_BASE+0*x*20	URFBx	UCC receive FIFO base	4 bytes	26.5.1/26-13	—
UCC*x*_BASE+0*x*24	URFSx	UCC receive FIFO size	2 bytes	26.5.2/26-13	—
UCC*x*_BASE+0*x*26	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0*x*28	URFETx	UCC receive FIFO emergency threshold	2 bytes	26.5.3/26-14	—
UCC*x*_BASE+0*x*2A	URFSETx	UCC receive FIFO special emergency threshold	2 bytes	26.5.4/26-14	—
UCC*x*_BASE+0*x*2C	UTFBx	UCC transmit FIFO base	4 bytes	26.5.5/26-15	—
UCC*x*_BASE+0*x*30	UTFSx	UCC transmit FIFO size	2 bytes	26.5.6/26-15	—
UCC*x*_BASE+0*x*32	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0x34	UTFETx	UCC transmit FIFO emergency threshold	2 bytes	26.5.7/26-16	—
UCC*x*_BASE+0x36	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0x38	UTFTT	UCC transmit FIFO transmit threshold	2 bytes	26.5.8/26-16	—
UCC*x*_BASE+0x3A	Reserved	—	2 bytes	—	—
UCC*x*_BASE+0x3C	UTPTx	UCC transmit polling timer	2 bytes	26.4.3/26-10	—
UCCx_BASE+0x3E	Reserved	—	2 bytes	—	—
UCCx_BASE+0x40	URTRYx	UCC retry counter register	4 bytes	26.5.9/26-17	—
UCC*x*_BASE+0x44 –UCC*x*_BASE+0x8F	Reserved	—	76 bytes	—	—
UCC*x*_BASE+0x90	GUEMRx	UCC general extended mode register	1 byte	22.3.2/22-5	—
UCCx_BASE+0x91 –UCCx_BASE+0xFF	Reserved	—	111 bytes	—	—
Ethernet General Configuration					
UCC*x*_BASE+0x100	MACCFG1	Mac configuration register #1	4 bytes	29.5.1.4/29-32	—
UCC*x*_BASE+0x104	MACCFG2	Mac configuration register #2	4 bytes	29.5.1.5/29-34	—
UCC*x*_BASE+0x108	IPGIFG	Interframe gap register	4 bytes	29.5.1.6/29-36	—
UCC*x*_BASE+0x10C	HAFDUP	Half-duplex register	4 bytes	29.5.1.7/29-37	—
UCC*x*_BASE+0x110	Reserved	—	12 bytes	—	—
UCC*x*_BASE+0x120	MIIMCFG	MII mgmt configuration register	4 bytes	29.5.1.8/29-38	—
UCC*x*_BASE+0x124	MIIMCOM	MII mgmt command register	4 bytes	29.5.1.9/29-39	—
UCC*x*_BASE+0x128	MIIMADD	MII mgmt address register	4 bytes	29.5.1.10/29-40	—
UCC*x*_BASE+0x12C	MIIMCON	MII mgmt control register	4 bytes	29.5.1.11/29-41	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UCC*x*_BASE+0x130	MIIMSTAT	MII mgmt status register	4 bytes	29.5.1.12/29-41	Read only
UCC*x*_BASE+0x134	MIIMIND	MII mgmt indication register	4 bytes	29.5.1.13/29-42	Read only
UCC*x*_BASE+0x138	Reserved	—	4 bytes	—	—
UCC*x*_BASE+0x13C	IFSTAT	Interface status register	4 bytes	29.5.1.14/29-43	Read only
UCC*x*_BASE+0x140	MACSTNADDR1	Station address part 1 register	4 bytes	29.5.1.15/29-43	—
UCCx_BASE+0x144	MACSTNADDR2	Station address part 2 register	4 bytes	29.5.1.16/29-44	—
UCC*x*_BASE+0x148	Reserved	—	8 bytes	—	—
UCC*x*_BASE+0x150	UEMPR	UCC Ethernet MAC parameter register	4 bytes	29.5.1.17/29-45	—
UCC*x*_BASE+0x154	Reserved	—	4 bytes	—	—
UCC*x*_BASE+0x158	UESCR	UCC Ethernet statistics control register	2 bytes	29.5.1.18/29-45	—
UCC*x*_BASE+0x15A–0x17F	Reserved	—	—	—	—
Ethernet Statistics Counters					
UCCx_BASE+0x180	TX64	Transmit and receive 64-byte frame counter	4 bytes	29.7.5/29-93	—
UCCx_BASE+0x184	TX127	Transmit and receive 65- to 127-byte frame counter	4 bytes	29.7.5/29-93	—
UCCx_BASE+0x188	TX255	Transmit and receive 128- to 255-byte frame counter	4 bytes	29.7.5/29-93	—
UCCx_BASE+0x18C	RX64	Receive and receive 64-byte frame counter	4 bytes	29.7.5/29-93	—
UCCx_BASE+0x190	RX127	Receive and receive 65- to 127-byte frame counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x194	Rx255	Receive and receive 128- to 255-byte frame counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x198	TXOK	Transmit good bytes counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x19C	TXCF	Transmit control frame counter	2 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1A0	TMCA	Transmit multicast control frame counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1A4	TBCA	Transmit broadcast packet counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1A8	RXFOK	Receive frame OK counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1B0	RBYT	Receive good and bad bytes counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1AC	RXBOK	Receive bytes OK counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1B4	RMCA	Receive multicast packet counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1B8	RBCA	Receive broadcast packet counter	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1BC	SCAR	Statistics carry register	4 bytes	29.7.5/29-93	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UCC*x*_BASE+0x1C0	SCAM	Statistics carry mask register	4 bytes	29.7.5/29-93	—
UCC*x*_BASE+0x1C4–0x1FF	Reserved	—	7 bytes	—	—
UTOPIA Controllers UTOPIA: UPC1_BASE=0x0_2E00					
UPCn_BASE+0x00	UPGCR	UTOPIA general configuration register	2 bytes	31.6.1/31-16	—
UPCn_BASE+0x04	UPLPA	UTOPIA last PHY address	2 bytes	31.6.2/31-17	—
UPCn_BASE+0x08	UPHEC	ATM HEC register	2 bytes	31.6.3/31-17	—
UPCn_BASE+0x0C	UPUC	UTOPIA UCC configuration	4 bytes	31.6.4/31-18	—
UPCn_BASE+0x10	UPDC1	UTOPIA device 1 configuration	4 bytes	31.6.5/31-19	—
UPCn_BASE+0x14	Reserved	—	4 bytes	—	—
UPCn_BASE+0x18	Reserved	—	4 bytes	—	—
UPCn_BASE+0x1C	Reserved	—	4 bytes	—	—
UPCn_BASE+0x20	Reserved	—	1 byte	—	—
UPCn_BASE+0x28	Reserved	—	4 bytes	—	—
UPCn_BASE+0x30	UPDRS1_H	UTOPIA device 1 rate select	4 bytes	31.6.9/31-22	—
UPCn_BASE+0x38	Reserved	—	4 bytes	—	—
UPCn_BASE+0x3C	Reserved	—	4 bytes	—	—
UPCn_BASE+0x40	Reserved	—	4 bytes	—	—
UPCn_BASE+0x44	Reserved	—	4 bytes	—	—
UPCn_BASE+0x48	Reserved	—	4 bytes	—	—
UPCn_BASE+0x4C	Reserved	—	4 bytes	—	—
UPCn_BASE+0x50	UPDRP1	UTOPIA device 1 receive priority low	4 bytes	31.6.10/31-23	—
UPCn_BASE+0x54	Reserved	—	4 bytes	—	—
UPCn_BASE+0x58	Reserved	—	4 bytes	—	—
UPCn_BASE+0x5C	Reserved	—	4 bytes	—	—
UPCn_BASE+0x60	UPDE1	UTOPIA device 1 event	4 bytes	31.6.11/31-23	—
UPCn_BASE+0x64	Reserved	—	4 bytes	—	—
UPCn_BASE+0x68	Reserved	—	4 bytes	—	—
UPCn_BASE+0x6C	Reserved	—	4 bytes	—	—
UPCn_BASE+0x70	UPRP1	UTOPIA device 1 internal rate config	2 bytes	31.6.6/31-21	—
UPCn_BASE+0x72	Reserved	—	2 bytes	—	—
UPCn_BASE+0x74	Reserved	—	2 bytes	—	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UPCn_BASE+0x76	Reserved	—	2 bytes	—	—
UPCn_BASE+0x80	UPTIRR1_0	Device 1 transmit internal rate 0	2 bytes	31.6.7/31-21	—
UPCn_BASE+0x82	UPTIRR1_1	Device 1 transmit internal rate 1	2 bytes		—
UPCn_BASE+0x84	UPTIRR1_2	Device 1 transmit internal rate 2	2 bytes		—
UPCn_BASE+0x86	UPTIRR1_3	Device 1 transmit internal rate 3	2 bytes		—
UPCn_BASE+0x88	Reserved	—	2 bytes	—	—
UPCn_BASE+0x8A	Reserved	—	2 bytes	—	—
UPCn_BASE+0x8C	Reserved	—	2 bytes	—	—
UPCn_BASE+0x8E	Reserved	—	2 bytes	—	—
UPCn_BASE+0x90	Reserved	—	2 bytes	—	—
UPCn_BASE+0x92	Reserved	—	2 bytes	—	—
UPCn_BASE+0x94	Reserved	—	2 bytes	—	—
UPCn_BASE+0x96	Reserved	—	2 bytes	—	—
UPCn_BASE+0x98	Reserved	—	2 bytes	—	—
UPCn_BASE+0x9A	Reserved	—	2 bytes	—	—
UPCn_BASE+0x9C	Reserved	—	2 bytes	—	—
UPCn_BASE+0x9E	Reserved	—	2 bytes	—	—
UPCn_BASE+0xA0	UPER1	Device 1 port enable register	4 bytes	31.6.8/31-22	—
UPCn_BASE+0xA4	Reserved	—	4 bytes	—	—
UPCn_BASE+0xA8	Reserved	—	4 bytes	—	—
UPCn_BASE+0xAC	Reserved	—	4 bytes	—	—
UPCn_BASE+0xB0–UPCn_BASE+0x1FF	Reserved	—	352 bytes	—	—
0x0_2800–0x0_2DFF, 0x0_3800–0x0_3DFF	Reserved	—	3 Kbytes	—	—
SDMA–General					
0x0_4000	SDSR	Serial DMA status register	4 bytes	18.1.8.1/18-6	—
0x0_4004	SDMR	Serial DMA mode register	4 bytes	18.1.8.2/18-7	—
0x0_4008	SDTR1	SDMA system bus threshold register	4 bytes	18.1.8.3/18-9	—
0x0_400C	SDTR2	SDMA secondary bus threshold register	4 bytes	18.1.8.3/18-9	—
0x0_4010	SDHY1	SDMA system bus hysteresis register	4 bytes	18.1.8.4/18-11	—
0x0_4014	SDHY2	SDMA secondary bus hysteresis register	4 bytes	18.1.8.4/18-11	—

Table 2-4. Detailed QUICC Engine Memory Map (continued)

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_4018	SDTA1	SDMA system bus address register	4 bytes	18.1.8.5/18-12	—
0x0_401C	SDTA2	SDMA secondary bus address register	4 bytes	18.1.8.5/18-12	—
0x0_4020	SDTM1	SDMA system bus MSNUM register	4 bytes	18.1.8.6/18-12	—
0x0_4024	SDTM2	SDMA secondary bus MSNUM register	4 bytes	18.1.8.6/18-12	—
0x0_4028–0x0_4037	Reserved	—	16 bytes	—	—
0x0_4038	SDAQR	SDMA address bus qualify register	4 bytes	18.1.8.7/18-13	—
0x0_403C	SDAQMR	SDMA address bus qualify mask register	4 bytes	18.1.8.8/18-14	—
0x0_4044	SDEBCR	SDMA CAM entries base register	4 bytes	18.1.8.9/18-14	—
0x0_4048–0x_0407F	Reserved	—	56 bytes	—	—
0x0_4080–0x0_42FF	Reserved	—	72 bytes	—	—
0x0_4300–0x0_43FF	Reserved	—	256 bytes	—	—
0x0_4400–-0x0_44FF	Reserved	—	256 bytes	—	—
0x0_4600–0x0_7FFF	Reserved	—	14.5 Kbytes	—	—
0x0_8000–0x3_FFFF	RAM Space				
0x0_8000–0x0_FFFF	Reserved	—	32 Kbytes	—	—
Multi-user RAM					
0x1_0000–0x1_3FFF	MURAM	Multi-user RAM	16 Kbytes	—	—
0x1_4000–0x1_BFFF	Reserved	—	32 Kbytes	—	—
0x1_C000–0x3_FFFF	Reserved	—	144 Kbytes	—	—
0x4_0000–0xF_FFFF	Reserved	—	768 Kbytes	—	—

Chapter 3
Signal Descriptions

This chapter describes the MPC8323E external signals. It is organized into the following sections:

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

NOTE

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

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

3.1 Signals Overview

The MPC8323E signals are grouped as follows:

- DDR memory interface signals
- PCI interface signals
- QUICC Engine interface signals
- DUART interface signals
- I^2C interface signals
- Local bus interface signals
- PIC interface signals
- PM interface signal
- JTAG, PMC, system control signals
- Clock signals

Figure 3-1 illustrates the external signals of the MPC8323E, showing how the signals are grouped. Refer to the *MPC8323E Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

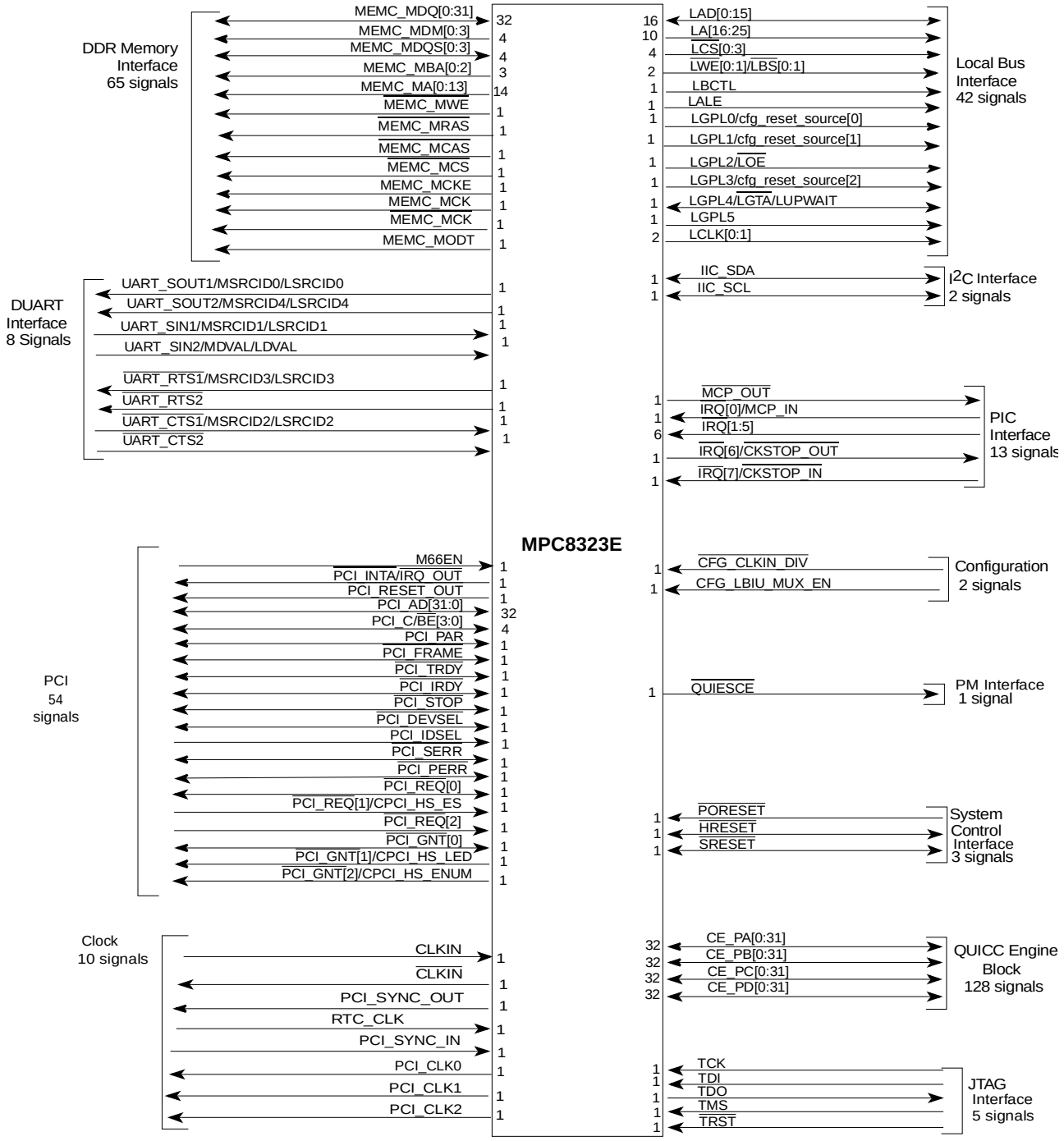


Figure 3-1. MPC8323E Signal Groupings

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

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

Table 3-1. MPC8323E Signal Reference by Functional Block

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
DDR SDRAM Memory Controller Interface						
MEMC_MDQ[0:31]	—	DDR data	DDR	32	I/O	9-3/9-5
MEMC_MDM[0:3]	—	DDR data mask 0–3	DDR	4	O	9-3/9-5
MEMC_MDQS[0:3]	—	DDR data strobe	DDR	4	I/O	9-3/9-5
MEMC_MBA[0:2]	—	DDR bank select	DDR	3	O	9-3/9-5
MEMC_MA[0:13]	—	DDR address	DDR	14	O	9-3/9-5
$\overline{\text{MEMC\_MWE}}$	—	DDR write enable	DDR	1	O	9-3/9-5
$\overline{\text{MEMC\_MRAS}}$	—	DDR row address strobe	DDR	1	O	9-3/9-5
$\overline{\text{MEMC\_MCAS}}$	—	DDR column address strobe	DDR	1	O	9-3/9-5
$\overline{\text{MEMC\_MCS}}$	—	DDR chip select (2/DIMM)	DDR	1	O	9-3/9-5
MEMC_MCKE	—	DDR clock enable	DDR	1	O	9-4/9-7
MEMC_MCK	—	DDR differential clocks (positive)	DDR	1	O	9-4/9-7
$\overline{\text{MEMC\_MCK}}$	—	DDR differential clocks (negative)	DDR	1	O	9-4/9-7
MEMC_MODT	—	On-die termination for DRAM chip	DDR	1	O	9-3/9-5
PCI						
$\overline{\text{PCI\_INTA}}$/ IRQ_OUT/	$\overline{\text{PCI\_INTA}}$	PCI interrupt output	PCI	1	O	13-3/13-5
	IRQ_OUT	Interrupt output	IPIC		O	13-3/13-5
$\overline{\text{PCI\_RESET\_OUT}}$	—	PCI reset	PCI	1	O	13-3/13-5
PCI_AD[31:0]	—	PCI address/data	PCI	32	I/O	13-3/13-5
PCI_C/$\overline{\text{BE}}$[3:0]	PCI_C	PCI command	PCI	4	I/O	13-3/13-5
	$\overline{\text{BE}}$[3:0]	Byte enable	PCI		I/O	13-3/13-5
PCI_PAR	—	PCI parity	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_FRAME}}$	—	PCI frame	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_TRDY}}$	—	PCI target ready	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_IRDY}}$	—	PCI initiator ready	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_STOP}}$	—	PCI stop	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_DEVSEL}}$	—	PCI device select	PCI	1	I/O	13-3/13-5
PCI_IDSEL	—	PCI initial device select	PCI	1	I	13-3/13-5

Table 3-1. MPC8323E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_SERR	—	PCI system error	PCI	1	I/O	13-3/13-5
PCI_PERR	—	PCI parity error	PCI	1	I/O	13-3/13-5
PCI_REQ0	—	PCI request 0	PCI	1	I/O	13-3/13-5
PCI_REQ1/ CPCI_HS_ES	PCI_REQ1	PCI request 1	PCI	1	I	13-3/13-5
	CPCI_HS_ES	Compact PCI hot swap ejector switch	CPCI		I	13-3/13-5
PCI_REQ2	—	PCI request 2	PCI	1	I	13-3/13-5
PCI_GNT0	—	PCI grant 0	PCI	1	I/O	13-3/13-5
PCI_GNT1/ CPCI_HS_LED	PCI_GNT1	PCI grant 1	PCI	1	O	13-3/13-5
	CPCI_HS_LED	Compact PCI hot swap LED	CPCI		O	13-3/13-5
PCI_GNT2/ CPCI_HS_ENUM	PCI_GNT2	PCI grant 2	PCI	1	O	13-3/13-5
	CPCI_HS_ENUM	Compact PCI hot swap enumerator	CPCI		O	13-3/13-5
M66EN	—	PCI 66-MHz timing on/off.	PCI	1	I	13-3/13-5
Local Bus Controller Interface						
LAD[0:15]	—	LBC address/data	LBC	16	I/O	10-2/10-5
LA[16:25]	—	LBC port address 16–25	LBC	10	O	10-2/10-5
LCS[0:3]	—	LBC chip select 0–3	LBC	4	O	10-2/10-5
LWE[0:1]/ LBS[0:1]	LWE[0:1]	LBC write enable 0–1	LBC	2	O	10-2/10-5
	LBS[0:1]	Byte select 0–1			O	10-2/10-5
LBCTL	—	LBC data buffer control	LBC	1	O	10-2/10-5
LALE	—	LBC address latch enable	LBC	1	O	10-2/10-5
LGPL0/cfg_reset_source0	LGPL0	LBC UPM general purpose line 0	LBC	1	O	10-2/10-5
	cfg_reset_source0	Configuration reset source 0	Reset & clock		I	4-1/4-1
LGPL1/cfg_reset_source1	LGPL1	LBC UPM general purpose line 1	LBC	1	O	10-2/10-5
	cfg_reset_source1	Configuration reset source 1	Reset & clock		I	4-1/4-1
LGPL2/LOE	LGPL2	LBC UPM general purpose line 2	LBC	1	O	10-2/10-5
	LOE	LBC output enable			O	10-2/10-5
LGPL3/cfg_reset_source2	LGPL3	LBC UPM general purpose line 3	LBC	—	O	10-2/10-5
	cfg_reset_source2	Configuration reset source 2	Reset & clock	—	I	4-1/4-1

Table 3-1. MPC8323E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
LGPL4/LGTA/ LUPWAIT	LGPL4	LBC UPM general purpose line 4	LBC	1	I/O	10-2/10-5
	LGTA	GPCM terminate access				
	LUPWAIT	UPM wait				
LGPL5	—	LBC UPM general purpose line 5	LBC	1	O	10-2/10-5
LCLK[0]	—	LBC clocks 0	LBC	1	O	10-2/10-5
LCLK[1]	—	LBC clocks 1	LBC	1	O	10-2/10-5
Programmable Interrupt Controller						
MCP_OUT	—	Machine check interrupt output	IPIC	1	O	8-2/8-5
IRQ[0]/ MCP_IN	IRQ[0]	External interrupt 0	IPIC	1	I	8-2/8-5
	MCP_IN	Machine check interrupt input	IPIC		I	8-2/8-5
IRQ[1]	—	External interrupt 1	IPIC	1	I	8-2/8-5
IRQ[2:5]	—	External interrupts 2–5	IPIC	4	I	8-2/8-5
IRQ[6]/ CKSTOP_OUT	IRQ[6]	External interrupt 6	IPIC	1	I	8-2/8-5
	CKSTOP_OUT	Checkstop output	CPU		O	4-3/4-4
IRQ[7]/ CKSTOP_IN	IRQ[7]	External interrupt 7	IPIC	1	I	8-2/8-5
	CKSTOP_IN	Checkstop input	CPU		I	4-3/4-4
DUART						
UART_SOUT1/ MSRCID[0]/ LSRCID[0]	UART_SOUT1	UART1 serial data out	UART	1	O	16-2/16-3
	MSRCID[0]	DDR memory debug source port ID 0	DDR		O	9-7/9-10
	LSRCID[0]	LBC debug source port ID 0	LBC		O	10-2/10-5
UART_SOUT2/ MSRCID[4]/ LSRCID[4]	UART_SOUT2	UART2 serial data out	UART	1	O	16-2/16-3
	MSRCID[4]	DDR memory debug source port ID 0	DDR		O	9-7/9-10
	LSRCID[4]	LBC debug source port ID 4	LBC		O	10-2/10-5
UART_SIN1/ MSRCID[1]/ LSRCID[1]	UART_SIN1	UART1 serial data in	UART	1	I	16-2/16-3
	MSRCID[1]	DDR memory debug source port ID 1	DDR		O	9-7/9-10
	LSRCID[1]	LBC debug source port ID 1	LBC		O	10-2/10-5
UART_SIN2/ MDVAL/ LDVAL	UART_SIN2	UART2 serial data in	UART	1	O	16-2/16-3
	MDVAL	DDR memory debug data valid	DDR		O	9-7/9-10
	LDVAL	LBC debug data valid	LBC		O	10-2/10-5

Table 3-1. MPC8323E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
UART_CTS1/ MSRCID[2]/ LSRCID[2]	UART_CTS1	UART1 clear to send	UART	1	I	16-2/16-3
	M1SRCID2	DDR memory debug source port ID 2	DDR		O	9-7/9-10
	LSRCID2	LBC debug source port ID 2	LBC		O	10-2/10-5
UART_CTS2	—	UART2 clear to send	UART	1	I	16-2/16-3
UART_RTS1 MSRCID[3]/ LSRCID[3]	UART_RTS1	UART1 ready to send	UART	1	O	16-2/16-3
	MSRCID3	DDR memory debug source port ID 3	DDR		O	9-7/9-10
	LSRCID3	LBC debug source port ID 3	LBC		O	10-2/10-5
UART_RTS2	—	UART2 ready to send	UART	1	O	16-2/16-3
I^2C Interface						
IIC_SDA	—	I2C serial data	I2C	1	I/O	15-2/15-3
IIC_SCL	—	I2C serial clock	I2C	1	I/O	15-2/15-3
QUICC Engine Block						
CE_PA[0:31]	—	QUICC Engine parallel port A	QUICC Engine	32	I/O	3-11/3-21
CE_PB[0:31]	—	QUICC Engine parallel port B	QUICC Engine	32	I/O	3-12/3-24
CE_PC[0:31]	—	QUICC Engine parallel port C	QUICC Engine	32	I/O	3-13/3-28
CE_PD[0:31]	—	QUICC Engine parallel port D	QUICC Engine	32	I/O	3-14/3-31
Clocks						
PCI_CLK_OUT[0:2]	—	PCI clock out 0–2	Clocks	3	O	4-2/4-3
CLKIN	—	Clock input	Clocks	1	I	4-2/4-3
CLKIN	—	Clock output	Clocks	1	O	4-2/4-3
RTC_CLK	—	—	Clocks	1	I	4-2/4-3
PCI_SYNC_IN	—	PCI clock sync input	Clocks	1	I	4-2/4-3
PCI_SYNC_OUT	—	PCI clock sync output	Clocks	1	O	4-2/4-3
JTAG						
TCK	—	Test clock	JTAG	1	I	17-2/17-2
TDI	—	Test data in	JTAG	1	I	17-2/17-2
TDO	—	Test data out	JTAG	1	O	17-2/17-2
TMS	—	Test mode select	JTAG	1	I	17-2/17-2

Table 3-1. MPC8323E Signal Reference by Functional Block (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
$\overline{\text{TRST}}$	—	Test reset	JTAG	1	I	17-2/17-2
PMC						
$\overline{\text{QUIESCE}}$	—	Quiesce state	PMC	1	O	5-60/5-60
Configuration						
$\overline{\text{CFG\_CLKIN\_DIV}}$	—	Configuration clock in division selection	Reset & clock	1	I	4-7/4-11
CFG_LBIU_MUX_EN	—	Selects multiplexing for address/data bus of the LBC block	Reset & clock	1	I	3-6/3-14
System Control						
$\overline{\text{PORESET}}$	—	Power-on reset	System control	1	I	4-1/4-1
$\overline{\text{HRESET}}$	—	Hard reset	System control	1	I/O	4-1/4-1
$\overline{\text{SRESET}}$	—	Soft reset	System control	1	I/O	4-1/4-1

Table 3-2 lists the signals in alphabetical order.

Table 3-2. MPC8323E Alphabetical Signal Reference

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
CE_PA[0:31]	—	QUICC Engine parallel port A	QUICC Engine	32	I/O	3-11/3-21
CE_PB[0:31]	—	QUICC Engine parallel port B	QUICC Engine	32	I/O	3-12/3-24
CE_PC[0:31]	—	QUICC Engine parallel port C	QUICC Engine	32	I/O	3-13/3-28
CE_PD[0:31]	—	QUICC Engine parallel port D	QUICC Engine	32	I/O	3-14/3-31
$\overline{\text{CFG\_CLKIN\_DIV}}$	—	Configuration clock in division selection	Reset & clock	1	I	4-7/4-11
CFG_LBIU_MUX_EN	—	Selects multiplexing for address/data bus of the LBC block	Reset & clock	1	I	3-6/3-14
CLKIN	—	Clock input	Clocks	1	I	4-2/4-3
$\overline{\text{CLKIN}}$	—	Clock output	Clocks	1	O	4-2/4-3
$\overline{\text{HRESET}}$	—	Hard reset	System control	1	I/O	4-4/4-5
IIC_SCL	—	I2C serial clock	I2C	1	I/O	15-2/15-3
IIC_SDA	—	I2C serial data	I2C	1	I/O	15-2/15-3
$\overline{\text{IRQ}}$[0]/ $\overline{\text{MCP\_IN}}$	$\overline{\text{IRQ}}$[0]	External interrupt 0	IPIC	1	I	8-2/8-5
	$\overline{\text{MCP\_IN}}$	Machine check interrupt input	IPIC		I	8-2/8-5
$\overline{\text{IRQ}}$[1]	—	External interrupt 1	IPIC	1	I	8-2/8-5
$\overline{\text{IRQ}}$[2:5]	—	External interrupts 2–5	IPIC	4	I	8-2/8-5
$\overline{\text{IRQ}}$[6]/ $\overline{\text{CKSTOP\_OUT}}$	$\overline{\text{IRQ}}$[6]	External interrupt 6	IPIC	1	I	8-2/8-5
	$\overline{\text{CKSTOP\_OUT}}$	Checkstop output	CPU		O	4-5/4-10
$\overline{\text{IRQ}}$[7]/ $\overline{\text{CKSTOP\_IN}}$	$\overline{\text{IRQ}}$[7]	External interrupt 7	IPIC	1	I	8-2/8-5
	$\overline{\text{CKSTOP\_IN}}$	Checkstop input	CPU		I	4-5/4-10
LA[16:25]	—	LBC port address 16–25	LBC	10	O	10-2/10-5
LAD[0:15]	—	LBC address/data	LBC	16	I/O	10-2/10-5
LALE	—	LBC address latch enable	LBC	1	O	10-2/10-5
LBCTL	—	LBC data buffer control	LBC	1	O	10-2/10-5
LCLK[0]	—	LBC clocks 0	LBC	1	O	10-2/10-5
LCLK[1]	—	LBC clocks 1	LBC	1	O	10-2/10-5
$\overline{\text{LCS}}$[0:3]	—	LBC chip select 0–3	LBC	4	O	10-2/10-5

Table 3-2. MPC8323E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
LGPL0/cfg_reset_source0	LGPL0	LBC UPM general purpose line 0	LBC	1	O	10-2/10-5
	cfg_reset_source0	Configuration reset source 0	Reset & clock		I	4-5/4-10
LGPL1/cfg_reset_source1	LGPL1	LBC UPM general purpose line 1	LBC	1	O	10-2/10-5
	cfg_reset_source1	Configuration reset source 1	Reset & clock		I	4-7/4-11
LGPL2/$\overline{\text{LOE}}$	LGPL2	LBC UPM general purpose line 2	LBC	1	O	10-2/10-5
	$\overline{\text{LOE}}$	LBC output enable			O	10-2/10-5
LGPL3/cfg_reset_source2	LGPL3	LBC UPM general purpose line 3	LBC		O	10-2/10-5
	cfg_reset_source2	Configuration reset source 2	Reset & clock		I	4-5/4-10
LGPL4/$\overline{\text{LGTA}}$/ LUPWAIT	LGPL4	LBC UPM general purpose line 4	LBC	1	I/O	10-2/10-5
	$\overline{\text{LGTA}}$	GPCM terminate access				
	LUPWAIT	UPM wait				
LGPL5	—	LBC UPM general purpose line 5	LBC	1	O	10-2/10-5
$\overline{\text{LWE}}$[0:1]/ $\overline{\text{LBS}}$[0:1]	$\overline{\text{LWE}}$[0:1]	LBC write enable 0–1	LBC	2	O	10-2/10-5
	$\overline{\text{LBS}}$[0:1]	Byte select 0–1			O	10-2/10-5
M66EN	—	PCI 66-MHz timing on/off.	PCI	1	I	13-3/13-5
$\overline{\text{MCP\_OUT}}$	—	Machine check interrupt output	IPIC	1	O	8-2/8-5
MEMC_MA[0:13]	—	DDR address	DDR	14	O	9-3/9-5
MEMC_MBA[0:2]	—	DDR bank select	DDR	3	O	9-3/9-5
$\overline{\text{MEMC\_MCAS}}$	—	DDR column address strobe	DDR	1	O	9-3/9-5
MEMC_MCK	—	DDR differential clocks (positive)	DDR	1	O	9-4/9-7
$\overline{\text{MEMC\_MCK}}$	—	DDR differential clocks (negative)	DDR	1	O	9-4/9-7
MEMC_MCKE	—	DDR clock enable	DDR	1	O	9-4/9-7
$\overline{\text{MEMC\_MCS}}$	—	DDR chip select (2/DIMM)	DDR	1	O	9-3/9-5
MEMC_MDM[0:3]	—	DDR data mask 0–3	DDR	4	O	9-3/9-5
MEMC_MDQ[0:31]	—	DDR data	DDR	32	I/O	9-3/9-5
MEMC_MDQS[0:3]	—	DDR data strobe	DDR	4	I/O	9-3/9-5
MEMC_MODT	—	On-die termination for DRAM chip	DDR	1	O	9-3/9-5
$\overline{\text{MEMC\_MRAS}}$	—	DDR row address strobe	DDR	1	O	9-3/9-5
$\overline{\text{MEMC\_MWE}}$	—	DDR write enable	DDR	1	O	9-3/9-5
$\overline{\text{MEMC\_MWE}}$	—	DDR write enable	DDR	1	O	9-3/9-5

Table 3-2. MPC8323E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_AD[31:0]	—	PCI address/data	PCI	32	I/O	13-3/13-5
PCI_C/$\overline{\text{BE}}$[3:0]	PCI_C	PCI command	PCI	4	I/O	13-3/13-5
	$\overline{\text{BE}}$[3:0]	Byte enable	PCI		I/O	13-3/13-5
PCI_CLK_OUT[0:2]	—	PCI clock out 0–2	Clocks	3	O	4-4/4-5
PCI_CLOCK/ PCI_SYNC_IN	PCI_CLOCK	PCI clock	Clocks	1	I	4-2/4-3
	PCI_SYNC_IN	PCI clock sync input	Clocks			
$\overline{\text{PCI\_DEVSEL}}$	—	PCI device select	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_FRAME}}$	—	PCI frame	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_GNT}}$[0]	—	PCI grant 0	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_GNT}}$[1]/ CPCI_HS_LED	$\overline{\text{PCI\_GNT}}$[1]	PCI grant 1	PCI	1	O	13-3/13-5
	CPCI_HS_LED	Compact PCI hot swap LED	CPCI		O	13-3/13-5
$\overline{\text{PCI\_GNT}}$[2]/ $\overline{\text{CPCI\_HS\_ENUM}}$	$\overline{\text{PCI\_GNT}}$[2]	PCI grant 2	PCI	1	O	13-3/13-5
	$\overline{\text{CPCI\_HS\_ENUM}}$	Compact PCI hot swap enumerator	CPCI		O	13-3/13-5
PCI_IDSEL	—	PCI initial device select	PCI	1	I	13-3/13-5
$\overline{\text{PCI\_INTA}}$/IRQ_OUT	$\overline{\text{PCI\_INTA}}$	PCI interrupt output	PCI	1	O	13-3/13-5
	IRQ_OUT	Interrupt output	IPIC		O	13-3/13-5
$\overline{\text{PCI\_IRDY}}$	—	PCI initiator ready	PCI	1	I/O	13-3/13-5
PCI_PAR	—	PCI parity	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_PERR}}$	—	PCI parity error	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_REQ}}$[0]	—	PCI request 0	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_REQ}}$[1]/ CPCI_HS_ES	$\overline{\text{PCI\_REQ}}$[1]	PCI request 1	PCI	1	I	13-3/13-5
	CPCI_HS_ES	Compact PCI hot swap ejector switch	CPCI		I	13-3/13-5
$\overline{\text{PCI\_REQ}}$[2]	—	PCI request 2	PCI	1	I	13-3/13-5
$\overline{\text{PCI\_RESET\_OUT}}$	—	PCI reset	PCI	1	O	13-3/13-5
$\overline{\text{PCI\_SERR}}$	—	PCI system error	PCI	1	I/O	13-3/13-5
$\overline{\text{PCI\_STOP}}$	—	PCI stop	PCI	1	I/O	13-3/13-5
PCI_SYNC_IN	—	PCI clock sync input	Clocks	1	I	4-4/4-5
PCI_SYNC_OUT	—	PCI clock sync output	Clocks	1	O	4-4/4-5
$\overline{\text{PCI\_TRDY}}$	—	PCI target ready	PCI	1	I/O	13-3/13-5
$\overline{\text{PORESET}}$	—	Power-on reset	System control	1	I	4-3/4-4

Table 3-2. MPC8323E Alphabetical Signal Reference (continued)

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
RTC_CLK	—		Clocks	1	I	4-4/4-5
$\overline{\text{QUIESCE}}$	—	Quiesce state	PMC	1	O	5-60/5-60
$\overline{\text{SRESET}}$	—	Soft reset	System control	1	I/O	4-3/4-4
TCK	—	Test clock	JTAG	1	I	17-2/17-2
TDI	—	Test data in	JTAG	1	I	17-2/17-2
TDO	—	Test data out	JTAG	1	O	17-2/17-2
TMS	—	Test mode select	JTAG	1	I	17-2/17-2
$\overline{\text{TRST}}$	—	Test reset	JTAG	1	I	17-2/17-2
$\overline{\text{UART\_CTS1}}$/ MSRCID[2]/ LSRCID[2]	$\overline{\text{UART\_CTS1}}$	UART1 clear to send	UART	1	I	16-2/16-3
	M1SRCID2	DDR memory debug source port ID 2	DDR		O	9-7/9-10
	LSRCID2	LBC debug source port ID 2	LBC		O	10-2/10-5
$\overline{\text{UART\_CTS2}}$	—	UART2 clear to send	UART	1	I	16-2/16-3
$\overline{\text{UART\_RTS1}}$ MSRCID[3]/ LSRCID[3]	$\overline{\text{UART\_RTS1}}$	UART1 ready to send	UART	1	O	16-2/16-3
	MSRCID3	DDR memory debug source port ID 3	DDR		O	9-7/9-10
	LSRCID3	LBC debug source port ID 3	LBC		O	10-2/10-5
$\overline{\text{UART\_RTS2}}$	—	UART2 ready to send	UART	1	O	16-2/16-3
UART_SIN1/ MSRCID[1]/ LSRCID[1]	UART_SIN1	UART1 serial data in	UART	1	I	16-2/16-3
	MSRCID[1]	DDR memory debug source port ID 1	DDR		O	9-7/9-10
	LSRCID[1]	LBC debug source port ID 1	LBC		O	10-2/10-5
UART_SIN2/ MDVAL/ LDVAL	UART_SIN2	UART2 serial data in	UART	1	O	16-2/16-3
	MDVAL		DDR		O	9-7/9-10
	LDVAL		LBC		O	10-2/10-5
UART_SOUT1/ MSRCID[0]/ LSRCID[0]	UART_SOUT1	UART1 serial data out	UART	1	O	16-2/16-3
	MSRCID[0]	DDR memory debug source port ID 0	DDR		O	9-7/9-10
	LSRCID[0]	LBC debug source port ID 0	LBC		O	10-2/10-5
UART_SOUT2/ MSRCID[4]/ LSRCID[4]	UART_SOUT2	UART2 serial data out	UART	1	O	16-2/16-3
	MSRCID[4]	DDR memory debug source port ID 0	DDR		O	9-7/9-10
	LSRCID[4]	LBC debug source port ID 4	LBC		O	10-2/10-5

3.2 Configuration Signals Sampled at Reset

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

Table 3-3. MPC8323E Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name
LBC	LGPL0	CFG_RESET_SOURCE0
	LGPL1	CFG_RESET_SOURCE1
	LGPL3	CFG_RESET_SOURCE2
Dedicated pin	None (dedicated pin)	$\overline{\text{CFG\_CLKIN\_DIV}}$
Dedicated pin	None (dedicated pin)	CFG_LBIU_MUX_EN

3.3 Output Signal States During Reset

When a system reset is recognized ($\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$ are asserted), the MPC8323E 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 MPC8323E ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state. Table 3-4 shows the states of the output-only signals.

Table 3-4. Output Signal States During System Reset

Interface	Signal	State During Reset
MEMC_MDM[0:3]	DDR data mask	All '0'
MEMC_MBA[0:2]	DDR bank select	All 'Z'
MEMC_MA[0:13]	DDR address	All 'Z'
$\overline{\text{MEMC\_MWE}}$	DDR write enable	'Z'
$\overline{\text{MEMC\_MRAS}}$	DDR row address strobe	'Z
$\overline{\text{MEMC\_MCAS}}$	DDR column address strobe	'Z'
$\overline{\text{MEMC\_MCS}}$	DDR chip select (2/DIMM)	'Z'
MEMC_MCKE	DDR clock enable	'0'
MEMC_MCK	DDR differential clock	'0'
$\overline{\text{MEMC\_MCK}}$	DDR differential clock	'0'
MEMC_MODT	DRAM on-die termination	'0'
$\overline{\text{INTA}}$/ $\overline{\text{IRQ\_OUT}}$	PCI interrupt output	High impedance
$\overline{\text{PCI\_RESET\_OUT}}$	PCI reset output	'0'

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

Interface	Signal	State During Reset
PCI_GNT[0:2]	PCI grant 0–2	High impedance
UART1_SOUT	UART1 serial data out	‘1’
UART1_RTS	UART1 ready to send	‘1’
LA[16:25]	LBC port address	Active – being used to load reset configuration word
LCS[0:3]	LBC chip select 1–4	‘1111’
LWE[0:1]/ LBS[0:1]	LBC write enable/byte select	‘11’ or ‘1111’
LBCTL	LBC data buffer control	Active – being used to load reset configuration word
LALE	LBC address latch enable	Active – being used to load reset configuration word
LGPL2/LOE	LBC output enable/GP line 2	Active – being used to load reset configuration word
LCLK[0:1]	LBC clock	‘0’
MCP_OUT	Machine check interrupt output	High impedance
TDO	Test data out	High impedance
QUIESCE	Quiesce state	‘1’
PCI_CLK_OUT[0:2]	PCI clock output 0–2	All ‘0’
PCI_SYNC_OUT	PCI sync output	Active clock

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

Table 3-5. Signals for Multiplexing

Signal Group	Muxing is Controlled By	Table/Page
CFG_LBIU_MUX_EN (LA[0:15]/CE_PA[16:31])	CFG_LBIU_MUX_EN	3-6/3-14
PCI/CPCI	RCWH[PCIARB]	4-12/4-14
QUICC Engine block	CPPARxA–CPPARxG	3-10/3-18
All others	SICRL	5-27/5-21, 5-27/5-21

3.4 Parallel I/O Ports

The QUICC Engine block supports 4 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. Each pin can be configured as open-drain (the pin can be configured in a wired-OR configuration on the board). The pin drives a zero voltage but is released to high impedance when driving a high voltage.

Note that port pins do not have an internal pull-up resistor. Due to the QUICC Engine block’s significant flexibility, many dedicated peripheral functions are multiplexed onto the ports. The functions are grouped to maximize the pins’ usefulness in the greatest number of MPC8323E applications. The reader may not

obtain a full understanding of the pin assignment capability described in this chapter without understanding the QUICC Engine peripherals.

3.4.1 Features

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

- Ports A, B, C, and D all have 32 pins.
- All ports are bidirectional.
- All ports have alternate on-chip peripheral functions.
- Each pin has four direction options: Input, Output, I/O and Disabled. A disabled pin is not driving any value and can be left floating outside the device (no need for external pull up or pull down).
- All port pins are disabled at hard reset.
- Usually pin values can be read while the pin is connected to an on-chip peripheral (this is not possible for port pins, which are used as PCI pins).
- Open-drain capability on all port pins
- Some of the port pins can be used as interrupt input pins.

3.4.2 External Signal Description

Table 3-6 provides a detailed description of the CFG_LBIU_MUX_EN (LA[0:15]) signals.

Table 3-6. External Signals—Detailed Signal Descriptions

Signal	I/O	Description	
CFG_LBIU_MUX_EN (LA[0:15]/CE_PA[16:31])	I	**State Meaning**	Asserted—QUICC Engine parallel I/O ports CE_PA[16:31] function according to the values in their configuration registers (this may be LBC or non-LBC functionality). Negated—QUICC Engine parallel I/O ports CE_PA[16:31] function as LA[0:15] irrespective of the value in their configuration registers.
		Timing	This signal should be connected to pull-up or pull-down on the board. The signal needs to be valid at all times and static.

3.4.3 Port Registers

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

The port registers are: CPODRx, CPDATx, CPDIR1x, CPDIR2x, CPPAR1x and CPPAR2x.

- The CPODRx registers (one bit per pin—bit number corresponds to pin number) determine the open-drain configuration for each pin.
- The CPDATx registers (one bit per pin—bit number corresponds to pin number) are used to read/write data from/to the pins.
- The CPDIR1x and CPDIR2x registers (two bits per pin) determine the in/out characteristics of the pin.

- The CPPAR1x and CPPAR2x registers (two bits per pin) determine the functionality of each pin, according to the respective pin assignment table.

3.4.3.1 QUICC Engine Port Open-Drain Registers (CPODRA–CPODRD)

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

Offset	0x1400 (CPODRA), 0x1418 (CPODRB), 0x1430 (CPODRC), 0x1448 (CPODRD)															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OD0	OD1	OD2	OD3	OD4	OD5	OD6	OD7	OD8	OD9	OD10	OD11	OD12	OD13	OD14	OD15
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	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
R/W	R/W															
Reset	0000_0000_0000_0000															

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

Table 3-7. CPODRx Field Descriptions

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

3.4.3.2 QUICC Engine Port Data Registers (CPDATA–CPDATD)

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

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

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

offset	0x1404 (CPDATA), 0x141C (CPDATB), 0x1434 (CPDATC), 0x144C (CPDATD)															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	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
R/W	R/W															
Reset	0000_0000_0000_0000															

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

Table 3-8. CPDATx Field Descriptions

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

3.4.3.3 QUICC Engine Port Direction Registers (CPDIRxA–CPDIRxD)

The QUICC Engine port direction registers determine the direction of each port pin. There are four possible direction options per port pin.

offset	0x1408 (CPDIR1A), 0x1420 (CPDIR1B), 0x1438 (CPDIR1C), 0x1450 (CPDIR1D)															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DIR0		DIR1		DIR2		DIR3		DIR4		DIR5		DIR6		DIR7	
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DIR8		DIR9		DIR10		DIR11		DIR12		DIR13		DIR14		DIR15	
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 3-4. QUICC Engine Port Direction 1 Registers (CPDIR1A–CPDIR1D)

offset	0x140C (CPDIR2A), 0x1424 (CPDIR2B), 0x143C (CPDIR2C), 0x1454 (CPDIR2D)															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DIR16		DIR17		DIR18		DIR19		DIR20		DIR21		DIR22		DIR23	
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DIR24		DIR25		DIR26		DIR27		DIR28		DIR29		DIR30		DIR31	
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 3-5. QUICC Engine Port Direction 2 Registers (CPDIR2A–CPDIR2D)

Table 3-9. CPDIRxy Field Descriptions

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

3.4.3.4 QUICC Engine Port Pin Assignment Registers (CPPARxA–CPPARxD)

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

offset	0x1410 (CPPAR1A), 0x1428 (CPPAR1B), 0x1440 (CPPAR1C), 0x1458 (CPPAR1D)															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SEL0		SEL1		SEL2		SEL3		SEL4		SEL5		SEL6		SEL7	
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SEL8		SEL9		SEL10		SEL11		SEL12		SEL13		SEL14		SEL15	
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 3-6. QUICC Engine Port Pin Assignment 1 Registers (CPPAR1A–CPPAR1D)

offset	0x1414 (CPPAR2A), 0x142C (CPPAR2B), 0x1444 (CPPAR2C), 0x145C (CPPAR2D)															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SEL16		SEL17		SEL18		SEL19		SEL20		SEL21		SEL22		SEL23	
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SEL24		SEL25		SEL26		SEL27		SEL28		SEL29		SEL30		SEL31	
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 3-7. QUICC Engine Port Pin Assignment 2 Registers (CPPAR2A–CPPAR2D)

Table 3-10. CPPARxy Field Descriptions

Bits	Name	Description
0–1, 2–3, etc.	SEL*n*	Determines the function of pin number *n*, according to Table 3-11 through Table 3-14 below. Functions not shown in the table represent Reserved values of these bits.

3.4.4 Port Block Diagram

Figure 3-8 shows the functional block diagram per one port pin.

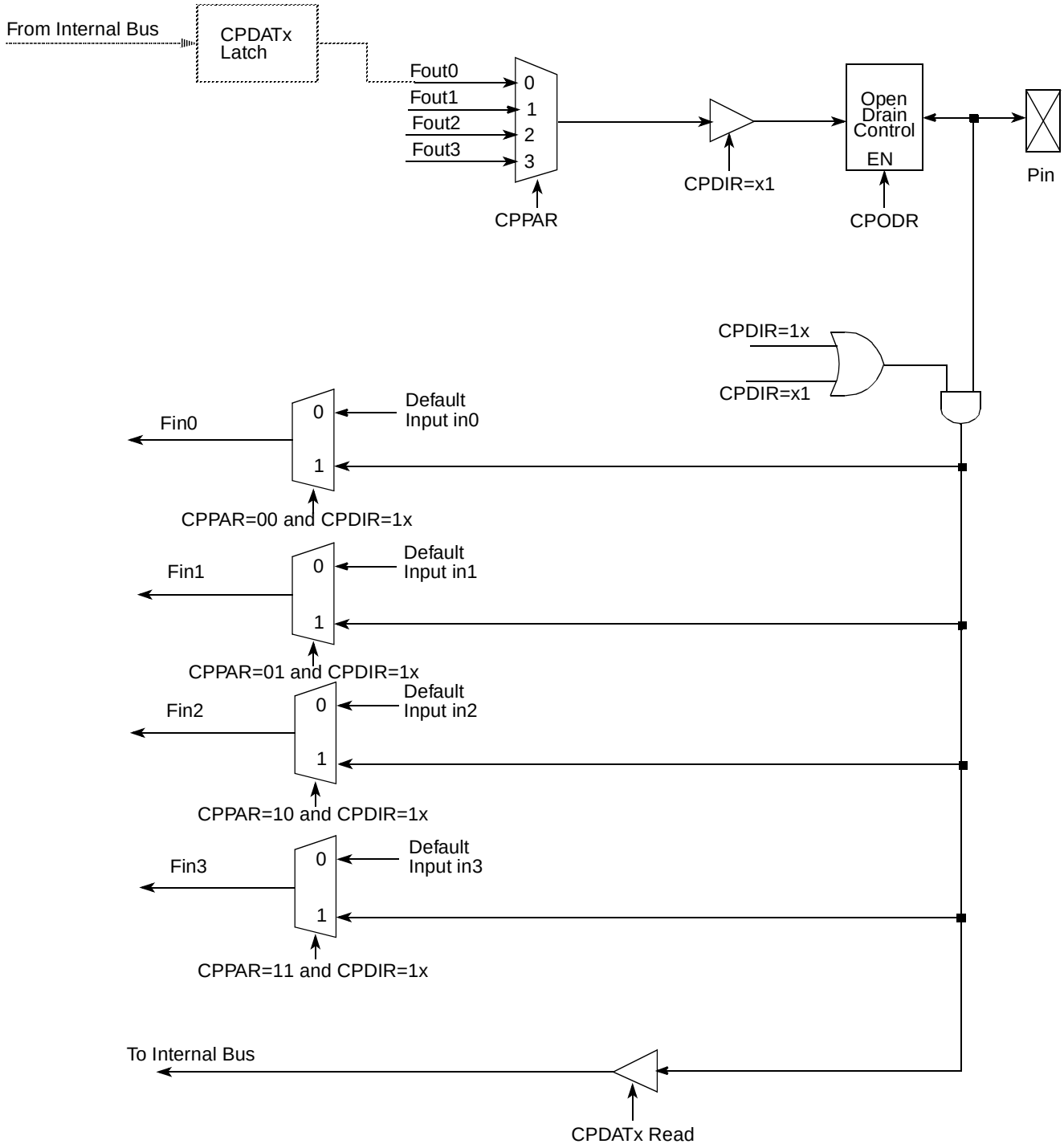


Figure 3-8. Port Functional Block Diagram

3.4.5 Port Pins Functions

Generally each pin can function as a general purpose I/O pin or as a dedicated input or output pin. Note however that some pins have only “General Purpose Input” or “General Purpose Output” function (not both possibilities). Refer to Table 3-11 through Table 3-14 for details. Usually following hard reset all port pins are disabled—both output and input buffers are off.

3.4.5.1 General-Purpose I/O Pins

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

3.4.5.2 Dedicated Pins

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

NOTE

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

Many input functions can also come from two or three different pins; see Section 3.4.7, “Ports Tables.”

3.4.6 QUICC Engine Ports Interrupts

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

See Section 8.6.10, “QUICC Engine Ports Interrupts,” for details on configuring these QUICC Engine ports interrupts.

3.4.7 Ports Tables

Table 3-11 through Table 3-14 describe the ports functionality according to the configuration of the port registers (CPPARxy and CPDIRxy).

In the tables below, for input functions that can come from two different pins, the default value for a primary option is simply a reference to the secondary option. In the secondary option, the programming is

relevant only if the primary option is not used for the function. A similar approach is used to describe input functions that can come from three different pins.

Table 3-11. Port A Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA0	IN	GPI_PA0	10	—	—	—	—	TDMA_TXD[0]/TDMA_TXD (Serial)	11	GND	—	—	—
	OUT	GPO_PA0	01	—	SER1_TXD[0]/SER1_TXD (Serial)	01	—				USBTXN	01	—
PA1	IN	GPI_PA1	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA1	01	—	SER1_TXD[1]	01	—	TDMA_TXD[1]	01	—	USBTXP	01	—
PA2	IN	GPI_PA2	10	—	—		—		—	—	—	—	—
	OUT	GPO_PA2	01	—	SER1_TXD[2]	01	—	TDMA_TXD[2]	01	—	—	—	—
PA3	IN	GPI_PA3	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA3	01	—	SER1_TXD[3]	01	—	TDMA_TXD[3]	01	—	—	—	—
PA4	IN	GPI_PA4	10	—	SER1_RXD[0]/SER1_RXD (Serial)	10	GND	TDMA_RXD[0]/TDMA_RXD (Serial)	11	GND	USBRXP	10	GND
	OUT	GPO_PA4	01	—	—	—	—				—	—	—
PA5	IN	GPI_PA5	10	—	SER1_RXD[1]	10	GND	TDMA_RXD[1]	10	GND	USBRXN	10	GND
	OUT	GPO_PA5	01	—	—	—	—	—	—	—	—	—	—
PA6	IN	GPI_PA6	10	—	SER1_RXD[2]	10	GND	TDMA_RXD[2]	10	GND	USBRXD	10	GND
	OUT	GPO_PA6	01	—	—	—	—	—	—	—	—	—	—
PA7	IN	GPI_PA7	10	—	SER1_RXD[3]	10	GND	TDMA_RXD[3]	10	GND	—	—	—
	OUT	GPO_PA7	01	—	—	—	—	—	—	—	—	—	—
PA8	IN	GPI_PA8	10	—	SER1_CD	10	GND	—	—	—	—	—	—
	OUT	GPO_PA8	01	—	—	—	—	TDMA_REQ	01		$\overline{\text{USBOE}}$	01	—
PA9	IN	GPI_PA9	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA9	01	—	—	—	—	TDMA_CLK	01	—	—	—	—
PA10	IN	GPI_PA10	10	—	SER1_CTS	10	GND	TDMA_RSYNC	10	GND	—	—	—
	OUT	GPO_PA10	01	—	—	—	—	—	—	—	—	—	—

Table 3-11. Port A Dedicated Pin Assignment (continued)

Pin	Direction	Pin Functions											
		CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA11	IN	GPI_PA11	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA11	01	—				TDMA_STROBE	01	—	—	—	—
PA12	IN	GPI_PA12	10	—	—	—	—	TDMA_TSYNC	10	GND	—	—	—
	OUT	GPO_PA12	01	—	SER1_RTS	01	—	—		—			
PA13	IN	GPI_PA13	10	—	CLK9	10	GND	—		—	—	—	—
	OUT	GPO_PA13	01	—	—	—	—	BRG9	01	—			
PA14	IN	GPI_PA14	10	—	CLK11	10	GND	—		—	—	—	—
	OUT	GPO_PA14	01	—	—	—	—	BRG10	01	—			
PA15	IN	GPI_PA15	10	—	—		—	—		—	—	—	—
	OUT	GPO_PA15	01	—				BRG7	01	—			
PA16	IN	GPI_PA16	10	—	—		—	—	—	—	—	—	—
	OUT	GPO_PA16	01	—				—	—	—	LA0 (Local Bus)	01	—
PA17	IN	GPI_PA17	10	—	—		—	—	—	—	—	—	—
	OUT	GPO_PA17	01	—				—	—	—	LA1 (Local Bus)	01	—
PA18	IN	GPI_PA18	10	—	—	—	—	TDMB_TXD[0]/TDMB_TXD (Serial)	11	GND	—	—	—
	OUT	GPO_PA18	01	—	ENET2_TXD[0]/SER2_TXD[0]/SER2_TXD (Serial)	01	—				LA2 (Local Bus)	01	—
PA19	IN	GPI_PA19	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA19	01	—	ENET2_TXD[1]/SER2_TXD[1]	01	—	TDMB_TXD[1]	01	—	LA3 (Local Bus)	01	—
PA20	IN	GPI_PA20	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA20	01	—	ENET2_TXD[2]/SER2_TXD[2]	01	—	TDMB_TXD[2]	01	—	LA4 (Local Bus)	01	—
PA21	IN	GPI_PA21	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PA21	01	—	ENET2_TXD[3]/SER2_TXD[3]	01	—	TDMB_TXD[3]	01	—	LA5 (Local Bus)	01	—

Table 3-11. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA22	IN	GPI_PA22	10	—	ENET2_RXD[0]/ SER2_RXD[0]/SER2_RXD (Serial)	10	GND	TDMB_RXD[0]/ TDMB_RXD (Serial)	11	GND	—	—	—
	OUT	GPO_PA22	01	—	—	—	—				LA6 (Local Bus)	01	—
PA23	IN	GPI_PA23	10	—	ENET2_RXD[1]/ SER2_RXD[1]	10	GND	TDMB_RXD[1]	10	GND	—	—	—
	OUT	GPO_PA23	01	—	—	—	—	—	—	—	LA7 (Local Bus)	01	—
PA24	IN	GPI_PA24	10	—	ENET2_RXD[2]/ SER2_RXD[2]	10	GND	TDMB_RXD[2]	10	GND	—	—	—
	OUT	GPO_PA24	01	—	—	—	—	—	—	—	LA8 (Local Bus)	01	—
PA25	IN	GPI_PA25	10	—	ENET2_RXD[3]/ SER2_RXD[3]	10	GND	TDMB_RXD[3]	10	GND	—	—	—
	OUT	GPO_PA25	01	—	—	—	—	—	—	—	LA9 (Local Bus)	01	—
PA26	IN	GPI_PA26	10	—	ENET2_RX_ER / SER2_CD	10	GND	—	—	—	—	—	—
	OUT	GPO_PA26	01	—	—	—	—	TDMB_REQ	01	—	LA10 (Local Bus)	01	—
PA27	IN	GPI_PA27	10	—	—	—	—	—	—	—			—
	OUT	GPO_PA27	01	—	ENET2_TX_ER	01	—	TDMB_CLK	01	—	LA11 (Local Bus)	01	—
PA28	IN	GPI_PA28	10	—	ENET2_RX_DV/ SER2_CTS	10	GND	TDMB_RSYNC	10	GND		—	—
	OUT	GPO_PA28	01	—	—	—	—	—	—	—	LA12 (Local Bus)	01	—
PA29	IN	GPI_PA29	10	—	ENET2_COL/SER2_RX	10	GND	—	—	—	—	—	—
	OUT	GPO_PA29	01	—	—	—	—	TDMB_STROBE	01	—	LA13 (Local Bus)	01	—
PA30	IN	GPI_PA30	10	—	—	—	—	TDMB_TSYNC	10	GND		—	—
	OUT	GPO_PA30	01	—	ENET2_TX_EN/ SER2_RTS	01	—	—	—	—	LA14 (Local Bus)	01	—

Table 3-11. Port A Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA31	IN	GPI_PA31	10	—	ENET2_CRS	10	GND	—	—	—	—	—	—
	OUT	GPO_PA31	01	—	—	—	—	—	—	—	LA15 (Local Bus)	01	—

Table 3-12. Port B Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PB0	IN	GPI_PB0	10	—	—	—	—	TDMC_TXD[0]/ TDMC_TXD (Serial)	11	GND	—	—	—
	OUT	GPO_PB0	01	—	ENET3_TXD[0]/ SER3_TXD[0]/SER3_TXD (Serial)	01	—						
PB1	IN	GPI_PB1	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB1	01	—	ENET3_TXD[1]/ SER3_TXD[1]	01	—	TDMC_TXD[1]	01	—			
PB2	IN	GPI_PB2	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB2	01	—	ENET3_TXD[2]/ SER3_TXD[2]	01	—	TDMC_TXD[2]	01	—			
PB3	IN	GPI_PB3	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB3	01	—	ENET3_TXD[3]/ SER3_TXD[3]	01	—	TDMC_TXD[3]	01	—			
PB4	IN	GPI_PB4	10	—	ENET3_RXD[0]/ SER3_RXD[0]/SER3_RXD (Serial)	10	GND	TDMC_RXD[0]/ TDMC_RXD (Serial)	11	GND	—	—	—
	OUT	GPO_PB4	01	—	—	—	—						

Table 3-12. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions											
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PB5	IN	GPI_PB5	10	—	ENET3_RXD[1]/ SER3_RXD[1]	10	GND	TDMC_RXD[1]	10	GND	—	—	—
	OUT	GPO_PB5	01	—	—	—	—	—	—	—			
PB6	IN	GPI_PB6	10	—	ENET3_RXD[2]/ SER3_RXD[2]	10	GND	TDMC_RXD[2]	10	GND	—	—	—
	OUT	GPO_PB6	01	—	—	—	—	—	—	—			
PB7	IN	GPI_PB7	10	—	ENET3_RXD[3]/ SER3_RXD[3]	10	GND	TDMC_RXD[3]	10	GND	—	—	—
	OUT	GPO_PB7	01	—	—	—	—	—	—	—	—	—	—
PB8	IN	GPI_PB8	10	—	ENET3_RX_ER / SER3_CD	10	GND	—	—	—	—	—	—
	OUT	GPO_PB8	01	—	—	—	—	TDMC_REQ	01	—	—	—	—
PB9	IN	GPI_PB9	10	—	—	—	—			—	—	—	—
	OUT	GPO_PB9	01	—	ENET3_TX_ER	01	—	TDMC_CLK	01	—	—	—	—
PB10	IN	GPI_PB10	10	—	ENET3_RX_DV/ SER3_CTS	10	GND	TDMC_RSYNC	10	GND	—	—	—
	OUT	GPO_PB10	01	—	—	—	—	—	—	—	—	—	—
PB11	IN	GPI_PB11	10	—	ENET3_COL/S ER3_RX	10	GND	—	—	—	—	—	—
	OUT	GPO_PB11	01	—	—	—	—	TDMC_STROB E	01	—	—	—	—
PB12	IN	GPI_PB12	10	—	—	—	—	TDMC_TSYNC	10	GND	—	—	—
	OUT	GPO_PB12	01	—	ENET3_TX_EN/ SER3_RTS	01	—	—		—	—	—	—
PB13	IN	GPI_PB13	10	—	ENET3_CRS	10	GND	—	—	—	—	—	—
	OUT	GPO_PB13	01	—	—	—	—	—	—	—	—	—	—
PB14	IN	GPI_PB14	10	—	CLK12	10	GND	—	—	—	—	—	—
	OUT	GPO_PB14	01	—	—	—	—	—	—	—	—	—	—

Table 3-12. Port B Dedicated Pin Assignment (continued)

Pin	Direction	Pin Functions											
		CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PB15	IN	GPI_PB15	10	—	—	—	—	—	—	—	UPC1_TXADDR[4]	11	GND
	OUT	GPO_PB15	01	—	—	—	—	—	—	—			
PB16	IN	GPI_PB16	10	—	—	—	—	—	—	—	UPC1_RXADDR[4]	11	GND
	OUT	GPO_PB16	01	—	—	—	—	—	—	—			
PB17	IN	GPI_PB17	10	—			—	CE_EXT_REQ1	10	GND	—	—	—
	OUT	GPO_PB17	01	—	BRG1	01	—	—	—	—	—	—	—
PB18	IN	GPI_PB18	10	—			—	TDMD_TXD[0]/ TDMD_TXD (Serial)	11	GND	—	—	—
	OUT	GPO_PB18	01	—	ENET4_TXD[0]/ SER4_TXD[0]/SER4_TXD (Serial)	01	—						
PB19	IN	GPI_PB19	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB19	01	—	ENET4_TXD[1]/ SER4_TXD[1]	01	—	TDMD_TXD[1]	01	—			
PB20	IN	GPI_PB20	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB20	01	—	ENET4_TXD[2]/ SER4_TXD[2]	01	—	TDMD_TXD[2]	01				
PB21	IN	GPI_PB21	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PB21	01	—	ENET4_TXD[3]/ SER4_TXD[3]	01	—	TDMD_TXD[3]	01	—			
PB22	IN	GPI_PB22	10	—	ENET4_RXD[0]/ SER4_RXD[0]/SER4_RXD (Serial)	10	GND	TDMD_RXD[0]/ TDMD_RXD (Serial)	11	GND	—	—	—
	OUT	GPO_PB22	01	—	—	—	—						
PB23	IN	GPI_PB23	10	—	ENET4_RXD[1]/ SER4_RXD[1]	10	GND	TDMD_RXD[1]	10	GND	—	—	—
	OUT	GPO_PB23	01	—	—	—	—	—	—	—			
PB24	IN	GPI_PB24	10	—	ENET4_RXD[2]/ SER4_RXD[2]	10	GND	TDMD_RXD[2]	10	GND	—	—	—
	OUT	GPO_PB24	01	—	—	—	—	—	—	—			

Table 3-12. Port B Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PB25	IN	GPI_PB25	10	—	ENET4_RXD[3]/ SER4_RXD[3]	10	GND	TDMD_RXD[3]	10	GND	—	—	—
	OUT	GPO_PB25	01	—	—	—	—	—	—	—	—	—	—
PB26	IN	GPI_PB26	10	—	ENET4_RX_ER / SER4_CD	10	GND	—	—	—	—	—	—
	OUT	GPO_PB26	01	—	—	—	—	TDMD_REQ	01	—	—	—	—
PB27	IN	GPI_PB27	10	—	—	—	—	—		—	—	—	—
	OUT	GPO_PB27	01	—	ENET4_TX_ER	01	—	TDMD_CLK	01	—	—	—	—
PB28	IN	GPI_PB28	10	—	ENET4_RX_DV/ SER4_CTS	10	GND	TDMD_RSYNC	10	GND	—	—	—
	OUT	GPO_PB28	01	—	—	—	—	—	—	—	—	—	—
PB29	IN	GPI_PB29	10	—	ENET4_COL/SER4_RX	10	GND		—	—	—	—	—
	OUT	GPO_PB29	01	—	—	—	—	TDMD_STROBE	01	—	—	—	—
PB30	IN	GPI_PB30	10	—	—	—	—	TDMD_TSYNC	10	GND	—	—	—
	OUT	GPO_PB30	01	—	ENET4_TX_EN/ SER4_RTS	01	—	—	—	—	—	—	—
PB31	IN	GPI_PB31	10	—	ENET4_CRS	10	GND	—	—	—	—	—	—
	OUT	GPO_PB31	01	—	—	—	—	—	—	—	—	—	—

Table 3-13. Port C Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PC0	IN	GPI_PC0	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC0	01	—	UPC1_TXDATA[0]	01	—	SER5_TXD[0]/SER5_TXD (Serial)	01	—	—	—	—
PC1	IN	GPI_PC1	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC1	01	—	UPC1_TXDATA[1]	01	—	SER5_TXD[1]	01	—	—	—	—
PC2	IN	GPI_PC2	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC2	01	—	UPC1_TXDATA[2]	01	—	SER5_TXD[2]	01	—	—	—	—
PC3	IN	GPI_PC3	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC3	01	—	UPC1_TXDATA[3]	01	—	SER5_TXD[3]	01	—	—	—	—
PC4	IN	GPI_PC4	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC4	01	—	UPC1_TXDATA[4]	01	—	—	—	—	—	—	—
PC5	IN	GPI_PC5	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC5	01	—	UPC1_TXDATA[5]	01	—	—	—	—	—	—	—
PC6	IN	GPI_PC6	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC6	01	—	UPC1_TXDATA[6]	01	—	—	—	—	—	—	—
PC7	IN	GPI_PC7	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC7	01	—	UPC1_TXDATA[7]	01	—	—	—	—	—	—	—
PC8	IN	GPI_PC8	10	—	UPC1_RXDATA[0]	10	GND	SER5_RXD[0]/SER5_RXD (Serial)	10	GND	—	—	—
	OUT	GPO_PC8	01	—	—	—	—	—	—	—	—	—	—

Table 3-13. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PC9	IN	GPI_PC9	10	—	UPC1_RXDATA[1]	10	GND	SER5_RXD[1]	10	GND	—	—	—
	OUT	GPO_PC9	01	—	—	—	—	—	—	—	—	—	—
PC10	IN	GPI_PC10	10	—	UPC1_RXDATA[2]	10	GND	SER5_RXD[2]	10	GND	—	—	—
	OUT	GPO_PC10	01	—	—	—	—	—	—	—	—	—	—
PC11	IN	GPI_PC11	10	—	UPC1_RXDATA[3]	10	GND	SER5_RXD[3]	10	GND			
	OUT	GPO_PC11	01	—	—	—	—	—	—	—	—	—	—
PC12	IN	GPI_PC12	10	—	UPC1_RXDATA[4]	10	GND	—	—	—	—	—	—
	OUT	GPO_PC12	01	—	—	—	—	—	—	—	—	—	
PC13	IN	GPI_PC13	10	—	UPC1_RXDATA[5]	10	GND		—	—	—	—	—
	OUT	GPO_PC13	01	—	—	—	—	LSRCID[0]	01	—			
PC14	IN	GPI_PC14	10	—	UPC1_RXDATA[6]	10	GND		—	—	—	—	—
	OUT	GPO_PC14	01	—	—	—	—	LSRCID[1]	01	—			
PC15	IN	GPI_PC15	10	—	UPC1_RXDATA[7]	10	GND		—	—	—	—	—
	OUT	GPO_PC15	01	—	—	—	—	LSRCID[2]	01	—			
PC16	IN	GPI_PC16	10	—	UPC1_TXADDR[0]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC16	01	—				—	—	—	—	—	—
PC17	IN	GPI_PC17	10	—	UPC1_TXADDR[1]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC17	01	—				LSRCID[3]	01	—			
PC18	IN	GPI_PC18	10	—	UPC1_TXADDR[2]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC18	01	—				LSRCID[4]	01	—			
PC19	IN	GPI_PC19	10	—	UPC1_TXADDR[3]	11	GND		—	—	—	—	—
	OUT	GPO_PC19	01	—				LDVAL	01	—			

Table 3-13. Port C Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PC20	IN	GPI_PC20	10	—	UPC1_RXADDR[0]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC20	01	—				—	—	—			
PC21	IN	GPI_PC21	10	—	UPC1_RXADDR[1]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC21	01	—				—	—	—			
PC22	IN	GPI_PC22	10	—	UPC1_RXADDR[2]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC22	01	—				—	—	—			
PC23	IN	GPI_PC23	10	—	UPC1_RXADDR[3]	11	GND	—	—	—	—	—	—
	OUT	GPO_PC23	01	—				—	—	—			
PC24	IN	GPI_PC24	10	—	UPC1_RXSOC	10	GND	SER5_CD	10	GND	—	—	—
	OUT	GPO_PC24	01	—	—	—	—	—	—	—	—	—	—
PC25	IN	GPI_PC25	10	—	UPC1_RXCLAV	11	GND	—	—	—	—	—	—
	OUT	GPO_PC25	01	—			—	—	—	—	—	—	—
PC26	IN	GPI_PC26	10	—	UPC1_RXPRTY	10	GND	CE_EXT_REQ2	10	GND	—	—	—
	OUT	GPO_PC26	01	—	—	—	—	—	—	—	—	—	—
PC27	IN	GPI_PC27	10	—	UPC1_RXEN	11	GND	—	—	—	—	—	—
	OUT	GPO_PC27	01	—				—	—	—	—	—	—
PC28	IN	GPI_PC28	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC28	01	—	UPC1_TXSOC	01	—	—	—	—	—	—	—
PC29	IN	GPI_PC29	10	—	UPC1_TXCLAV	11	GND	SER5_CTS	10	GND	—	—	—
	OUT	GPO_PC29	01	—				—	—	—	—	—	—
PC30	IN	GPI_PC30	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PC30	01	—	UPC1_TXPRTY	01	—	—	—	—	—	—	—
PC31	IN	GPI_PC31	10	—	UPC1_TXEN	11	GND	—	—	—	—	—	—
	OUT	GPO_PC31	01	—				SER5_RTS	01	—	—	—	—

Table 3-14. Port D Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PD0	IN	GPI_PD0	10	—	SPI1_MOSI	11	GND	—		—	—	—	—
	OUT	GPO_PD0	01	—						—	—	—	—
PD1	IN	GPI_PD1	10	—	SPI1_MISO	11	GND	—		—	—	—	—
	OUT	GPO_PD1	01	—						—	—	—	—
PD2	IN	GPI_PD2	10	—	SPI1_CLK	11	GND	—	—	—	—	—	—
	OUT	GPO_PD2	01	—					—	—	—	—	—
PD3	IN	GPI_PD3	10	—	SPI1_SEL	10	GND	—	—	—	—	—	—
	OUT	GPO_PD3	01	—	—	—	—		—	—	—	—	—
PD4	IN	GPI_PD4	10	—	SPI2_MDIO	11	GND	CE_MUX_MDIO	11	GND	—	—	—
	OUT	GPO_PD4	01	—							—	—	—
PD5	IN	GPI_PD5	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PD5	01	—	SPI2_MDC	01	—	CE_MUX_MDC	01	—	—	—	—
PD6	IN	GPI_PD6	10	—	CLK8	10	GND	—	—	—	CE_EXT_REQ3	10	GND
	OUT	GPO_PD6	01	—	—	—	—	BRG16	01	—	—	—	—
PD7	IN	GPI_PD7	10	—	GTM1_TIN1/ GTM2_TIN2	10	GND	—	—	—	CLK5	10	GND
	OUT	GPO_PD7	01	—	—	—	—				—	—	—
PD8	IN	GPI_PD8	10	—	GTM1_TGATE1 / GTM2_TGATE2	10	—	—	—	—	CLK6	10	GND
	OUT	GPO_PD8	01	—	—	—	—				—	—	—
PD9	IN	GPI_PD9	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PD9	01	—	GTM1_TOUT1	01	—						
PD10	IN	GPI_PD10	10	—	GTM1_TIN2/ GTM2_TIN1	10	GND	—	—	—	CLK17	10	GND
	OUT	GPO_PD10	01	—	—	—	—				—	—	—

Table 3-14. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PD11	IN	GPI_PD11	10	—	GTM1_TGATE2 / GTM2_TGATE1	10	—	—	—	—	—	—	—
	OUT	GPO_PD11	01	—	—	—	—				—	01	—
PD12	IN	GPI_PD12	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PD12	01	—	GTM1_TOUT2	01	—	$\overline{\text{GTM2\_TOUT1}}$	01	—			
PD13	IN	GPI_PD13	10	—	GTM1_TIN3/ GTM2_TIN4	10	GND	—	—	—	—	—	—
	OUT	GPO_PD13	01	—	—	—	—				BRG8	01	—
PD14	IN	GPI_PD14	10	—	GTM1_TGATE3 / GTM2_TGATE4	10	—	—	—	—	—	—	—
	OUT	GPO_PD14	01	—	—	—	—						
PD15	IN	GPI_PD15	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PD15	01	—	GTM1_TOUT3	01	—						
PD16	IN	GPI_PD16	10	—	GTM1_TIN4/ GTM2_TIN3	10	GND	—	—	—	—	—	—
	OUT	GPO_PD16	01	—	—	—	—	—	—	—			
PD17	IN	GPI_PD17	10	—	GTM1_TGATE4 / GTM2_TGATE3	10	—	—	—	—	—	—	—
	OUT	GPO_PD17	01	—	—	—	—	—	—	—			
PD18	IN	GPI_PD18	10	—	—	—	—	—	—	—	—	—	—
	OUT	GPO_PD18	01	—	GTM1_TOUT4	01	—	$\overline{\text{GTM2\_TOUT3}}$	01	—			
PD19	IN	GPI_PD19	10	—	CE_RISC1_INT	10	GND	CE_EXT_REQ4	10	GND	—	—	—
	OUT	GPO_PD19	01	—	—	—	—	—	—	—			
PD20	IN	GPI_PD20	10	—	CLK18	10	GND	—	—	—	—		—
	OUT	GPO_PD20	01	—	—	—	—	BRG6	01	—	—		—

Table 3-14. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PD21	IN	GPI_PD21	10	—	CLK16	10	GND	—	—	—	—		—
	OUT	GPO_PD21	01	—	—	—	—	BRG5	01	—	UPC1_CLK	01	—
PD22	IN	GPI_PD22	10	—	CLK4	10	GND	—	—	—	—	—	—
	OUT	GPO_PD22	01	—	—	—	—	BRG9	01	—	UCC2_CLK	01	—
PD23	IN	GPI_PD23	10	—	CLK3	10	GND	—	—	—	—	—	—
	OUT	GPO_PD23	01	—	—	—	—	BRG10	01	—	UCC3_CLK	01	—
PD24	IN	GPI_PD24	10	—	CLK10	10	GND	—	—	—	—	—	—
	OUT	GPO_PD24	01	—	—	—	—	BRG2	01	—	UCC4_CLK	01	—
PD25	IN	GPI_PD25	10	—	CLK13	10	GND	—	—	—	—	—	—
	OUT	GPO_PD25	01	—	—	—	—	BRG16	01	—	UCC5_CLK	01	—
PD26	IN	GPI_PD26	10	—	CLK2	10	GND	—	—	—	—	—	—
	OUT	GPO_PD26	01	—	—	—	—	BRG4	01	—	UCC1_CLK	01	—
PD27	IN	GPI_PD27	10	—	CLK1	10	GND	—	—	—	—	—	—
	OUT	GPO_PD27	01	—	—	—	—	BRG3	01	—	—	—	—
PD28	IN	GPI_PD28	10	—	CLK19	10	GND	—	—	—	—	—	—
	OUT	GPO_PD28	01	—	—	—	—	BRG11	01	—	—	—	—
PD29	IN	GPI_PD29	10	—	CLK15	10	GND	—	—	—	—	—	—
	OUT	GPO_PD29	01	—	—	—	—	BRG8	01	—	—	—	—
PD30	IN	GPI_PD30	10	—	CLK14	10	GND	—	—	—	—	—	—
	OUT	GPO_PD30	01	—	—	—	—		01	—	—	—	—
PD31	IN	GPI_PD31	10	—	CLK7	10	GND	—	—	—	—	—	—
	OUT	GPO_PD31	01	—	—	—	—	BRG15	01	—	—	—	—

Chapter 4
Reset, Clocking, and Initialization

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

4.1 External Signals

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

4.1.1 Reset Signals

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

Table 4-1. System Control Signals

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

Table 4-1. System Control Signals (continued)

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

4.1.2 Clock Signals

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

Table 4-2. External Clock Signals

Signal	I/O	Description	
CLKIN	I	System clock. In PCI host mode, CLKIN is the primary input clock. CLKIN directly feeds the PCI output clock dividers and is driven out on the PCI_SYNC_OUT signal for de-skewing external PCI clocks routing. In PCI agent mode, this signal should be tied to GND. When the device is in PCI host mode but the PCI_CLK_OUT[0:2] signals are not driven, PCI clock distribution should be done externally and CLKIN should be tied to GND.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Requirements	Should be tied low in PCI agent mode.
		Reset State	Always input.
$\overline{\text{CLKIN}}$	O	Crystal output. CLKIN/$\overline{\text{CLKIN}}$ allows the system clock to be provided through an external oscillator.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Requirements	Should be left unconnected if unused.
		Reset State	Always output.
PCI_CLK/ PCI_SYNC_IN	I	PCI clock/ PCI synchronization clock (PCI_CLK/PCI_SYNC_IN). In PCI agent mode or in PCI host mode when the PCI_CLK_OUT[0:2] signals are not driven, PCI_CLK is the primary clock input to the device. In PCI host mode with PCI_CLK_OUT[0:2] driven, PCI_SYNC_IN is connected externally to PCI_SYNC_OUT	
		Timing	Assertion/Negation—See the hardware specifications for timing information
		Reset State	Always input.
PCI_SYNC_OUT	O	Reference PCI output synchronization clock (PCI_SYNC_OUT). In PCI host mode with the PCI_CLK_OUT[0:2] signals driven, PCI_SYNC_OUT is connected externally to PCI_SYNC_IN signal for de-skewing external PCI clocks routing. PCI_SYNC_OUT has the same frequency as CLKIN or CLKIN/2 depending on the state of $\overline{\text{CFG\_CLKIN\_DIV}}$ at reset. See Section 4.3.1.2, "CLKIN Division." In PCI agent mode, this signal is typically not used.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Reset State	Always output, toggling in PCI host mode.
PCI_CLK_OUT[0:2]	O	PCI output clocks bank. In PCI host mode, the device provides three separate clock output signals for feeding PCI agent devices.	
		Timing	Assertion/Negation—See the hardware specifications for timing information.
		Reset State	Always output. Drive '0' and after power-on reset flow. Enabled by a memory-mapped register.

4.2 Functional Description

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

4.2.1 Reset Operations

The device has several inputs to the reset logic:

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

All of these reset sources are fed into the reset controller and, depending on the source of the reset, different actions are taken. The reset status register, described in Section 4.5.1.3, "Reset Status Register (RSR)," indicates the last sources to cause a reset.

4.2.1.1 Reset Causes

Table 4-3 describes reset causes.

Table 4-3. Reset Causes

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

Table 4-3. Reset Causes (continued)

Name	Description
Software hard reset	A hard reset sequence can be initialized by writing to a memory-mapped register (RCR).
Software soft reset	A soft reset sequence can be initialized by writing to a memory-mapped register (RCR).

4.2.1.2 Reset Actions

The reset control logic determines the cause of reset, synchronizes it if necessary, and resets the appropriate internal hardware. Each reset flow has a different impact on the device logic:

- Power-on reset has the greatest impact, resetting the entire device, including clock logic and error capture registers.
- Hard reset resets the entire device excluding clock logic and error capture registers.
- Soft reset initializes the internal logic while maintaining the system configuration.

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

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

Table 4-4. Reset Actions

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

4.2.2 Power-On Reset Flow

Assertion of the PORESET external signal initiates the power-on reset flow. PORESET should be asserted externally for at least 32 input clock cycles after stable external power to the device is applied.

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

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

1. Power is applied to meet the specifications in the device data sheet.
2. The system asserts PORESET and TRST, causing all registers to be initialized to their default states and most I/O drivers to be released to high-impedance.

   Some clock, clock enabled, and system control signals remain active.
3. The system applies a stable CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) signal and stable reset configuration inputs (CFG_RESET_SOURCE, CFG_CLKIN_DIV).
4. The system negates PORESET after at least 32 stable CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) clock cycles.
5. The device samples the reset configuration input signals to determine the clock division and the reset configuration words source.
6. The device starts loading the reset configuration words.

   Loading time depends on the reset configuration word source.
7. When the reset configuration word low is loaded, the system PLL and QUICC Engine PLL begin lock.

   When the system PLL is locked, *csb_clk* is supplied to the core PLL.
8. The core PLL begins to lock.
9. The device drives HRESET asserted until the e300 PLL is locked and the reset configuration words are loaded.
10. The user optionally negates HRESET if it was not negated earlier.

    JTAG logic must always be initialized by asserting TRST. If the JTAG signals are not used, TRST should be connected directly to PORESET. TRST must not remain asserted after the negation of PORESET. There is no need to assert the SRESET signal when HRESET is asserted.
11. The internal reset to the core and the rest of the logic is negated. I/O drivers are enabled. The PCI interface can assert DEVSEL in response to configuration cycles.
12. The device stops driving SRESET and SRESET is negated. The reset to the e300 core is negated and the core is enabled.The boot sequencer, if enabled, is released, causing it to load configuration data from serial ROMs, as described in Section 15.4.5, “Boot Sequencer Mode.”

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

The device is now in its ready state.

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

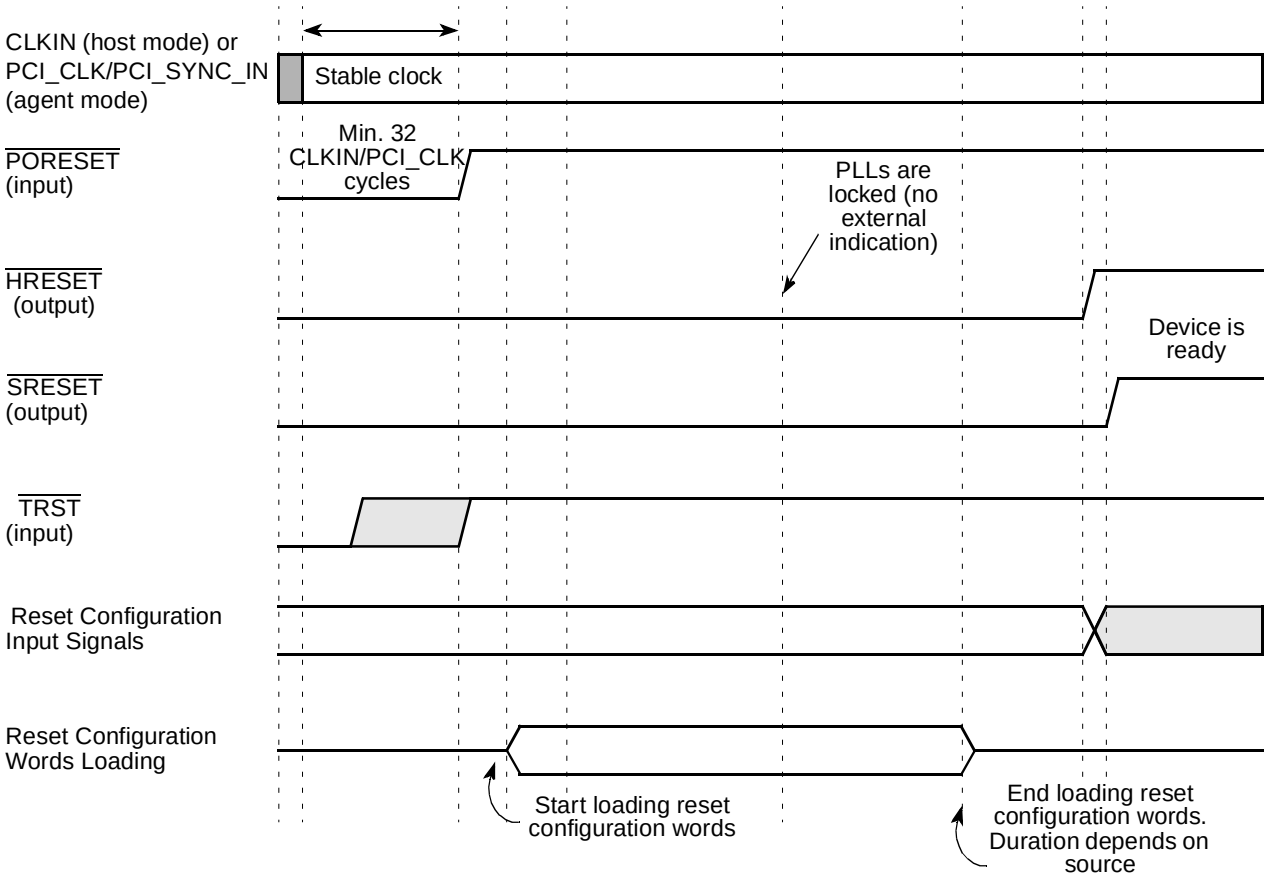


Figure 4-1. Power-On Reset Flow

4.2.3 Hard Reset Flow

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

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

NOTE

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

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

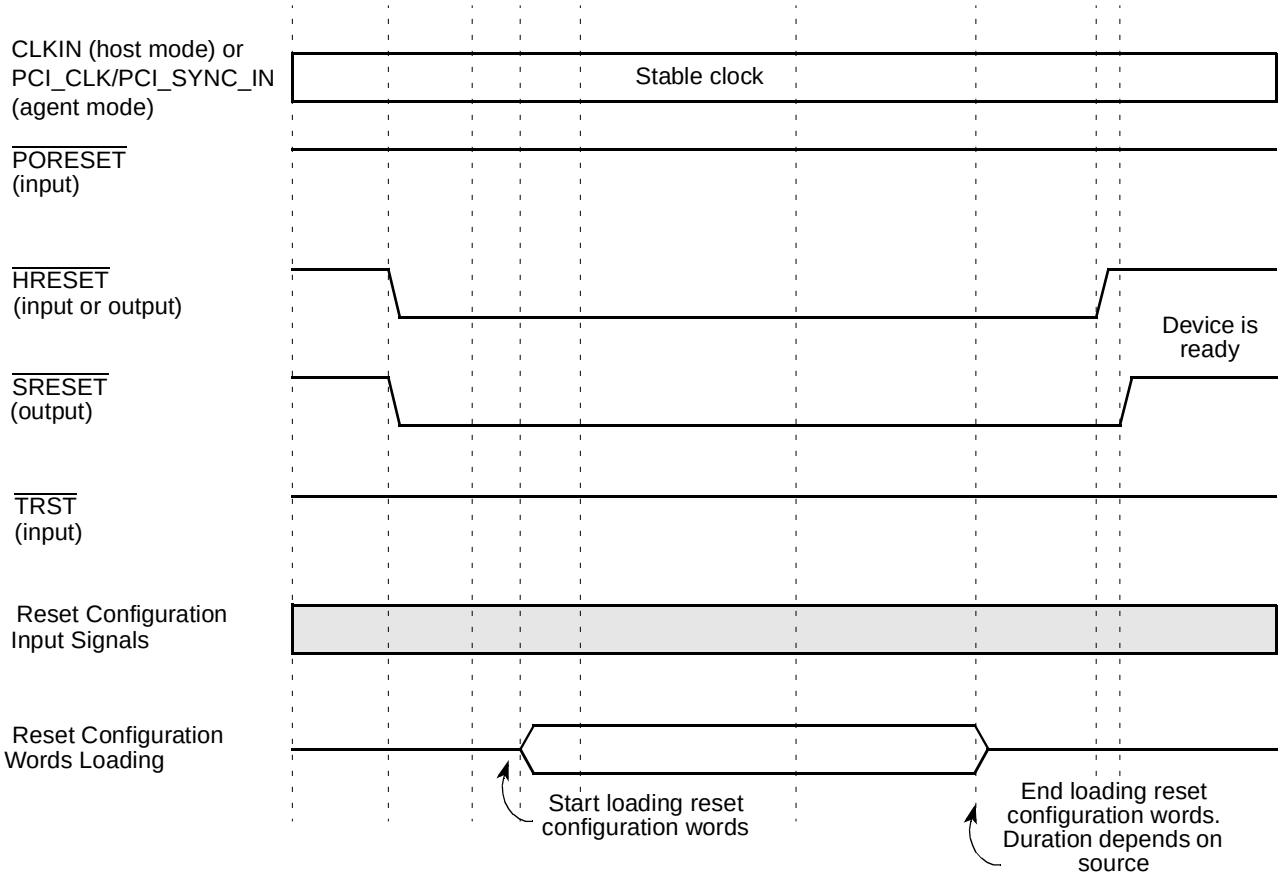


Figure 4-2. Hard Reset Flow

4.2.4 Soft Reset Flow

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

presence of an external (hard/soft) reset. While SRESET is asserted, internal hardware is reset but hard reset configuration does not change.

4.3 Reset Configuration

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

4.3.1 Reset Configuration Signals

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

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

NOTE

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

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

4.3.1.1 Reset Configuration Word Source

The reset configuration word source options, shown in Table 4-5, select whether the device loads a reset configuration word from a local bus EEPROM or I^2C EEPROM or uses hard-coded default options. The value of these signals also affects the duration of power-on and hard reset sequences.

Table 4-5. Reset Configuration Words Source

CFG_RESET_SOURCE[0:2]	Meaning
000	Reset configuration word is loaded from a local bus EEPROM
001	Reserved, should be cleared.
010	Reset configuration word is loaded from an I^2C EEPROM. PCI_CLK is valid for any PCI frequency up to 66.666 MHz (range of 24–66.666 MHz).
011	Hard-coded option 0. Reset configuration word is not loaded.
100	Hard-coded option 1. Reset configuration word is not loaded.
101	Hard-coded option 2. Reset configuration word is not loaded.
110	Hard-coded option 3. Reset configuration word is not loaded.
111	Hard-coded option 4. Reset configuration word is not loaded.

4.3.1.2 CLKIN Division

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

When the device is configured as a PCI agent, the $\overline{\text{CFG\_CLKIN\_DIV}}$ configuration input can be used to double the internal clock frequencies, if sampled as ‘0’ during power-on reset assertion. This feature is useful if a fixed internal frequency is desired regardless of whether the PCI clock is running at 33 or 66 MHz. PCI specifications require the PCI clock frequency information to be provided by the M66EN signal.

When the device is configured as PCI host, there are two scenarios for connecting the $\overline{\text{CFG\_CLKIN\_DIV}}$ configuration input. If the frequency of CLKIN is 33 MHz (that is, the PCI system is running on a 33-MHz clock), $\overline{\text{CFG\_CLKIN\_DIV}}$ should be connected high. If the frequency of CLKIN is 66 MHz (that is, the PCI system can run at 33- or 66-MHz clock signaled by M66EN), $\overline{\text{CFG\_CLKIN\_DIV}}$ should be connected to the M66EN signal.

Table 4-6. Division

$\overline{\text{CFG\_CLKIN\_DIV}}$	Description
1	In PCI host mode,: PCI_SYNC_OUT = 1:1 and all PCI_CLK_OUT[0:2] clocks are running at a frequency which is equal to the frequency
0	In PCI agent mode,: PCI_SYNC_OUT = 2:1 and all PCI_CLK_OUT[0:2] clocks are running at a frequency which is equal to the PCI_SYNC_OUT frequency.

4.3.1.3 Selecting Reset Configuration Input Signals

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

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

Table 4-7. Selecting Reset Configuration Input Signals

I^2C EEPROM Configuration Words	CLKIN Frequency (Host Mode)	$\overline{\text{CFG\_CLKIN\_DIV}}$ (Host Mode)	PCI_CLK Frequency (Agent Mode)	CFG_RESET_SOURCE[0:2]	Reset Sequence Duration in CLKIN/ PCI_CLK Cycles	Duration
No	33 MHz	1	33 MHz	000, 011–111 (not I^2C EEPROM)	15380	462 µs
No	66 MHz	1	66 MHz	000, 011–111 (not I^2C EEPROM)	15380	231 µs
No	66 MHz	0	33 MHz	000, 011–111 (not I^2C EEPROM)	30760/15380	462 µs
Yes	33 MHz	1	33 MHz	010 (I^2C EEPROM)	75816	1048 µs
Yes	66 MHz	1	66 MHz	010 (I^2C EEPROM)	37908	568 µs
Yes	66 MHz	0	33 MHz	010 (I^2C EEPROM)	—	1048 µs

4.3.2 Reset Configuration Words

The reset configuration words control the clock ratios and other basic device functions such as PCI host or agent mode, boot location, and endian mode. The reset configuration words are loaded from the local bus or from the I^2C interface or from hard-coded values during the power-on or hard reset flows. See Section 4.3.1, “Reset Configuration Signals,” for information on the reset configuration word source. Although the configuration reset words are loaded during hard reset flows, the clocks and PLL modes are reset only when $\overline{\text{PORESET}}$ is asserted during a power-on reset flow. See Section 4.2.1.2, “Reset Actions.” The values of fields in the reset configuration words registers (RCWLR and RCWHR) reflect only their state during the reset flow. Some of these parameters and modes can be modified by changing their values in the memory-mapped registers of other units, which does not affect RCWLR and RCWHR.

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

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

See Section 4.5, “Memory Map/Register Definitions.”

4.3.2.1 Reset Configuration Word Low Register (RCWLR)

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

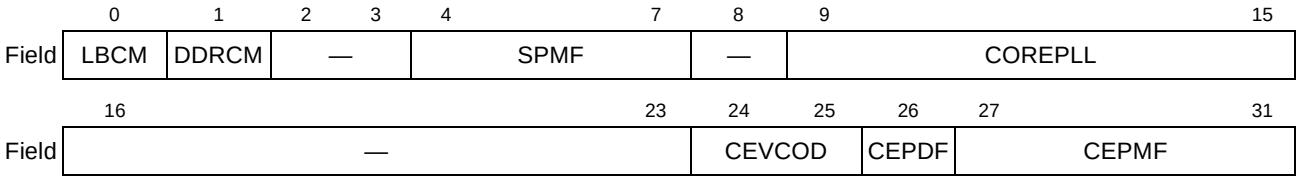


Figure 4-3. Reset Configuration Word Low Register (RCWLR)

Table 4-8 defines the RCWLR bit fields.

Table 4-8. RCWLR Bit Settings

Bits	Name	Description
0	LBCM	Local bus memory controller clock mode. Must be cleared. Selects the local bus controller clock ratio. The local bus memory controller operates with a frequency equal to the frequency of *csb_clk*. This bit should be cleared. 0 *csb_clk* ratio is 1:1 Reserved
1	DDRCM	DDR SDRAM memory controller clock mode. Selects the DDR SDRAM memory controller clock ratio. The DDR SDRAM memory controller operates at twice the frequency of the *csb_clk*. DDRCM must be set. Reserved 1 *csb_clk* ratio is 2:1
2–3	—	Reserved. Must be configured as 2'b10.
4–7	SPMF	System PLL multiplication factor.
8	—	Reserved, should be cleared
9–15	COREPLL	Core PLL configuration. COREPLL sets the ratio between the e300 core clock and the internal *csb_clk* of the device. The encodings for COREPLL are given in the hardware specifications for this device.
16–23	—	Reserved, should be cleared.
24–25	CEVCOD	QUICC Engine PLL VCO division. Establishes the internal ratio between the QUICC Engine PLL VCO point and the QUICC Engine PLL output. 00 4 01 Reserved, should be cleared. 10 2 11 Reserved, should be cleared **Notes**:Set CEPDF to 0 always. Set CEVCOD to 00 (Division factor of 4) for QE frequency below 150 MHz. Set CEVCOD to 10 (Division factor of 2) for QE frequency above 150 MHz.
26	CEPDF	QUICC Engine PLL division factor. Selects whether the PLL output is halved. By setting this bit, non-integer ratios between the primary clock input and *ce_clk* can be achieved. 0 *ce_clk* = primary clock input × CEPMF 1 *ce_clk* = (primary clock input × CEPMF)/ 2
27–31	CEPMF	QUICC Engine PLL multiplication factor. See Section 4.3.2.1.2, "QUICC Engine PLL Multiplication Factor," for more information

4.3.2.1.1 System PLL Configuration

The system PLL ratio reset, shown in Table 4-9, establishes the clock ratio between the CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) input signal and the internal *csb_clk* of the device. *csb_clk* drives internal units and feeds the e300 core PLL. The supported range of system clock (CLKIN/PCI_SYNC_IN) input frequency is 25–66 MHz.

Table 4-9. System PLL Ratio

RCWLR Bits	Field Name	Value (Binary)	***csb_clk*** **: CLKIN (PCI Host Mode)** ***csb_clk*** **: (PCI_CLK x (1+~sampled_$\overline{\text{cfg_clkin_div}}$))** **(PCI Agent Mode)**
4–7	SPMF	0000	Reserved
		0001	Reserved
		0010	2 : 1
		0011	3 : 1
		0100	4 : 1
		0101	5 : 1
		0110	6 : 1
		0111–1111	Reserved, should not be set

NOTE

In PCI host mode, the SPMF field described in Table 4-9 always selects the *csb_clk*:CLKIN ratio regardless of the $\overline{\text{CFG\_CLKIN\_DIV}}$ reset configuration input value during reset flow.

The SPMF field maximum allowed value is dependent on the value sampled on $\overline{\text{CFG\_CLKIN\_DIV}}$ during power-on reset. Table 4-10 defines the upper limit of SPMF with respect to these values. Values for SPMF are as follows:

Table 4-10. SPMF Maximum Values

$\overline{\text{CFG\_CLKIN\_DIV}}$	LBCM	DDRCM	Maximum SPMF Value (decimal)
1	0	1	6
0	0	1	3

4.3.2.1.2 QUICC Engine PLL Multiplication Factor

The RCWLR field CEPMF (QUICC Engine PLL multiplication factor), shown in Table 4-11, along with the QUICC Engine PLL division factor (CEPDF,) establishes the ratio between *ce_clk* and the primary input clock.

Table 4-11. QUICC Engine PLL Multiplication Factor

RCWLR Bits	Field Name	Value (Binary)	QUICC Engine Block PLL Multiplication Factor
27–31	CEPMF	00000	Reserved, should be cleared.
		00001	Reserved, should be cleared.
		00010	2
		00011	3
		00100	4
		00101	5
		00110	6
		00111	7
		01000	8
		01001–11111	Reserved, should be cleared.

4.3.2.2 Reset Configuration Word High Register (RCWHR)

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

Offset 0x0_0904 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	11	12	15
Field	PCIHOST	—	PCIARB	—	COREDIS	BMS	BOOTSEQ		SWEN	ROMLOC		—	—

	16	27	28	29	30	31
Field	—		TLE	LALE	—	

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

Table 4-12 defines the reset configuration word high bit fields.

Table 4-12. Reset Configuration Word High Bit Settings

Bits	Name	Description
0	PCIHOST	PCI host mode. See Section 4.3.2.2.1, “PCI Host/Agent Configuration,” for more information.
1	—	Reserved, should be cleared.

Table 4-12. Reset Configuration Word High Bit Settings (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2</td><td>PCIARB</td><td>PCI internal arbiter mode. Enables the on-chip PCI arbiter.
0 On-chip PCI arbiter is disabled. External arbitration is required.
1 On-chip PCI arbiter is enabled.
The value of PCIARB also defines the function of the PCI arbitration signals that are multiplexed with CompactPCI signals, as follows:
<table>
<tr><th>Pin Function When PCIARB = 0</th><th>Pin Function When PCIARB = 1</th></tr>
<tr><td>CPCI_HS_ES</td><td>$\overline{\text{PCI\_REQ}}$[1]</td></tr>
<tr><td>CPCI_HS_LED</td><td>$\overline{\text{PCI\_GNT}}$[1]</td></tr>
<tr><td>CPCI_HS_ENUM</td><td>$\overline{\text{PCI\_GNT}}$[2]</td></tr>
</table>
</td></tr>
<tr><td>3</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>4</td><td>COREDIS</td><td>Core disable mode. Specifies the e300 core mode out of reset. If COREDIS is set, the core cannot fetch boot code until it is configured by an external master. The external master frees the core to boot by clearing the COREDIS bit in the arbiter configuration register as described in Section 6.2.1, “Arbiter Configuration Register (ACR).”
This bit must be set when the boot sequencer is enabled to initiate the device (BOOTSEQ is not 0b00). Otherwise, unpredictable behavior occurs.
0 The core can boot without waiting for configuration by an external master.
1 Core boot holdoff mode. The core is prevented from booting until it is configured by an external master.</td></tr>
<tr><td>5</td><td>BMS</td><td>Boot memory space.
See Section 4.3.2.2.2, “Boot Memory Space (BMS),” for more information.</td></tr>
<tr><td>6–7</td><td>BOOTSEQ</td><td>Boot sequencer configuration.
See Section 4.3.2.2.3, “Boot Sequencer Configuration,” for more information.</td></tr>
<tr><td>8</td><td>SWEN</td><td>Software watchdog enable. Selects whether the software watchdog is enabled to start counting down immediately when coming out of reset. The user can override this value by writing to the system watchdog control register (SWCRR[SWEN]) during system initialization.
0 Disabled
1 Enabled</td></tr>
<tr><td>9–11</td><td>ROMLOC</td><td>Boot ROM interface location.
See Section 4.3.2.2.4, “Boot ROM Location,” for more information.</td></tr>
<tr><td>12–15</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>16–27</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>28</td><td>TLE</td><td>True little-endian. See Section 4.3.2.2.5, “e300 Core True Little-Endian,” for more information.</td></tr>
<tr><td>29</td><td>LALE</td><td>Local bus LALE signal timing. See Section 4.3.2.2.6, “LALE Configuration,” for more information.</td></tr>
</table>

4.3.2.2.1 PCI Host/Agent Configuration

The PCIHOST configuration parameter, shown in Table 4-13, configures the device to act as a PCI host or as a PCI agent device. In host mode, the device can immediately master transactions to the PCI interface. If the device is a PCI agent device, the device is disabled from mastering PCI transactions until the external

host enables it to do so. The external host does this by setting the control registers of the device's interfaces appropriately. See details in the PCI programming model described in Section 13.3, "Memory Map/Register Definitions."

Table 4-13. PCI Host/Agent Configuration

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

NOTE

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

4.3.2.2.2 Boot Memory Space (BMS)

BMS defines the initial value of the e300 core MSR[IP] bit, which specifies the location of the interrupt vectors (including the hard reset exception vector). The device defines the default boot ROM memory space to be 8 Mbytes at addresses 0x0000_0000 to 0x007F_FFFF or 0xFF80_0000 to 0xFFFF_FFFF. When the core comes out of reset, if it is enabled to boot, it begins fetching boot code from one of two addresses: 0x0000_0100 or 0xFFF0_0100, and exceptions are vectored to physical addresses 0x000*n_nnnn* or 0xFFF*n_nnnn* appropriately. This bit specifies whether an interrupt vector offset is prepended with 0xFFF or 0x000. In the description below, *n_nnnn* is the offset of the exception vector.

The boot memory space reset configuration word field, shown in Table 4-14, specifies both the device boot ROM address window and the initial e300 core boot address.

Table 4-14. Boot Memory Space

RCWHR Bit	Field Name	Value (Binary)	Meaning
5	BMS	0	Boot memory space is 8 Mbytes at 0x0000_0000 to 0x007F_FFFF. e300 core register MSR[IP] initial value is 0b0. The core, if enabled to boot, begins fetching boot code from address 0x0000_0100 and exceptions are vectored to the physical address of 0x000*n_nnnn*.
		1	Boot memory space is 8 Mbytes at 0xFF80_0000 to 0xFFFF_FFFF. e300 core register MSR[IP] initial value is 0b1. The core, if enabled to boot, begins fetching boot code from address 0xFFF0_0100 and exceptions are vectored to the physical address of 0xFFF*n_nnnn*.

4.3.2.2.3 Boot Sequencer Configuration

The boot sequencer configuration options, shown in Table 4-15, 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 device. These options also specify normal or extended I^2C addressing modes. See Section 15.4.5, "Boot Sequencer Mode."

Table 4-15. Boot Sequencer Configuration

RCWHR Bits	Field Name	Value (Binary)	Meaning
6–7	BOOTSEQ	00	Boot sequencer is disabled. No I^2C ROM is accessed.
		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	Reserved, should be cleared.

NOTE

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

4.3.2.2.4 Boot ROM Location

The device defines the default boot ROM address range to be 8 Mbytes at addresses 0x0000_0000 to 0x007F_FFFF or 0xFF80_0000 to 0xFFFF_FFFF (selected by the BMS reset configuration word field). However, the on-chip peripheral that manages these boot ROM accesses can be selected at power up.

The boot ROM location reset configuration word field, shown in Table 4-16, establishes the location of boot ROM. Accesses to the boot vector and the default boot ROM region of the local address map are directed to the interface specified by this field.

Table 4-16. Boot ROM Location

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

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

NOTE

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

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

4.3.2.2.5 e300 Core True Little-Endian

The true little endian reset configuration word field, shown in Table 4-17, selects whether the e300 core operates in big-endian mode or true little-endian mode at reset.

Table 4-17. e300 Core True Little-Endian

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

4.3.2.2.6 LALE Configuration

The LALE reset configuration word field, shown in Table 4-18, configures the timing of the local bus LALE signal. Refer to the hardware specifications for specific timing information.

Table 4-18. LALE Configuration

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

4.3.3 Loading the Reset Configuration Words

The device loads the reset configuration words from a local bus EEPROM or an I^2C serial EEPROM, or uses hard-coded configuration, as selected by the reset configuration inputs described in Section 4.3.1, “Reset Configuration Signals.” The following sections describe each of these options.

4.3.3.1 Loading from Local Bus EEPROM

The reset configuration words are assumed to reside in an EEPROM device connected to $\overline{\text{LCS0}}$ of the device local bus. Because the port size of this EEPROM is unknown, the device reads all configuration words byte-by-byte only from locations that are independent of port size.

Table 4-19 shows addresses that should be used to contain the reset configuration words. Byte addresses that do not appear in this table have no effect on the configuration of the device. The values of the bytes in Table 4-19 are always read on byte lane LAD[0:7] regardless of the port size.

Table 4-19. Local Bus Configuration EEPROM Addresses

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

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

Table 4-20 shows the data structure of the local bus device containing the reset configuration words (RCWL and RCWH).

Table 4-20. Local Bus Reset Configuration Words Data Structure

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

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

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

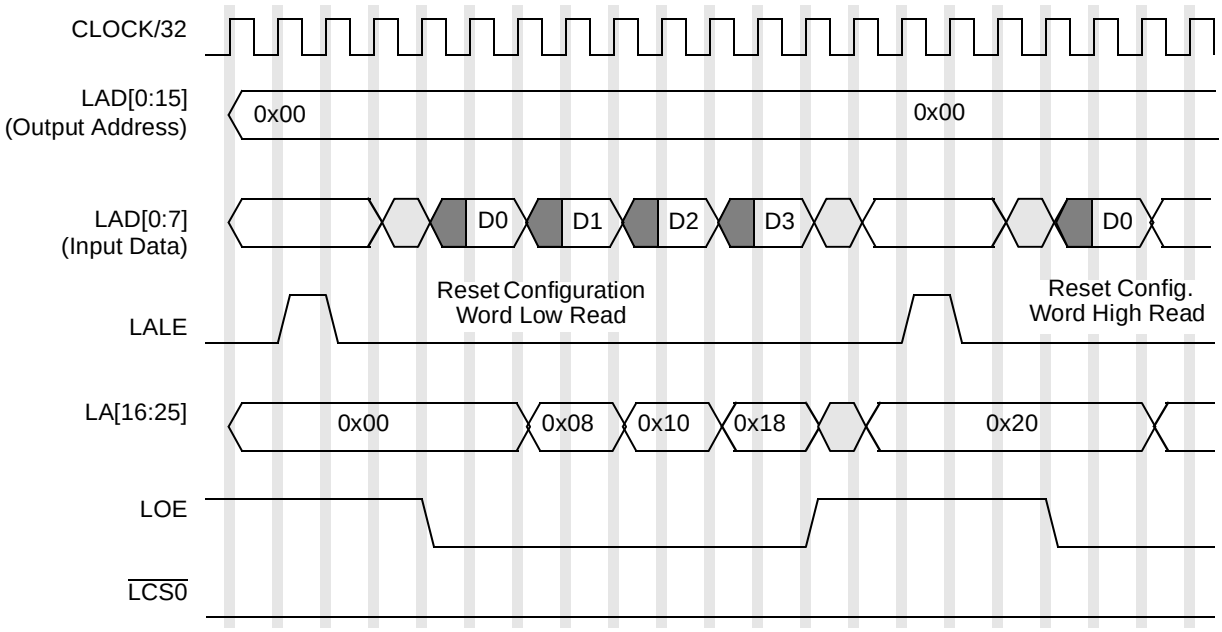


Figure 4-5. Loading Reset Configuration Words from Local Bus

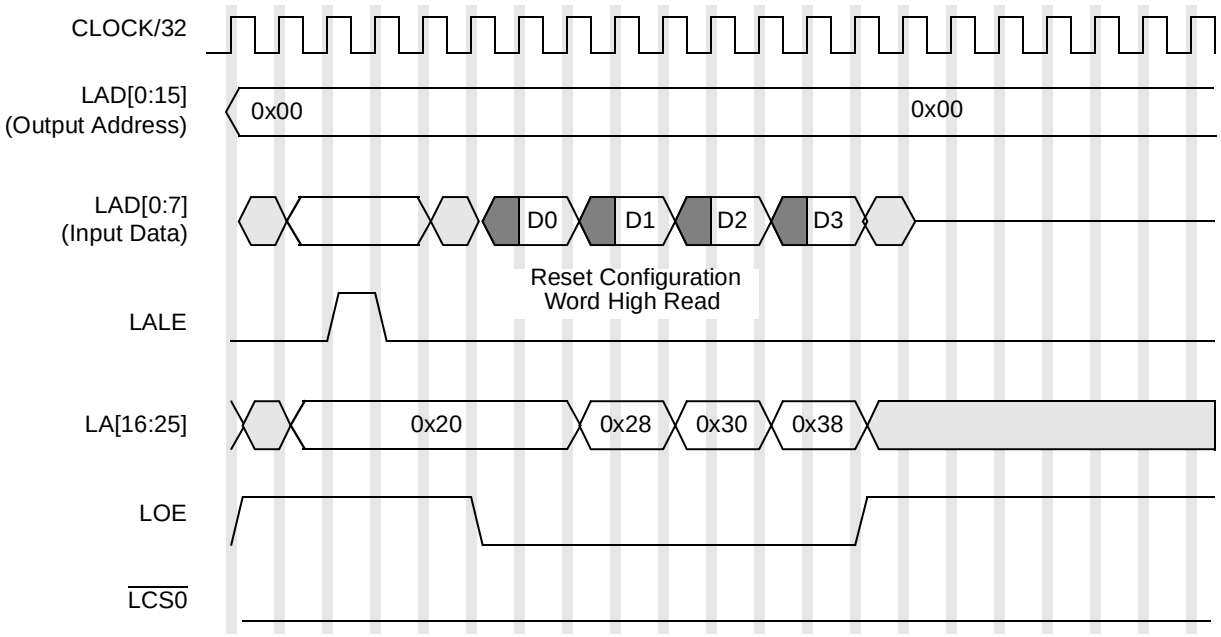


Figure 4-6. Loading Reset Configuration Words from Local Bus (continued)

4.3.3.2 Loading from I²C EEPROM

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

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

4.3.3.2.1 Using the Boot Sequencer Reset Configuration

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

NOTE

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

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

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

4.3.3.2.2 EEPROM Calling Address

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

4.3.3.2.3 EEPROM Data Format in Reset Configuration Mode

The I^2C 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 I^2C module checks to ensure that this preamble is correctly detected before proceeding further. Following the preamble, there should be the two reset configuration words, programmed according to a particular format, as shown in Figure 4-7.

The first 3 bytes hold the attributes and address offset. The addresses of the two reset configuration words must be programmed to the offset of the reset configuration word low register (RCWLR) and reset configuration word high register (RCWHR) respectively (see Section 4.5.1.1, “Reset Configuration Word Low Register (RCWLR),” and Section 4.5.1.2, “Reset Configuration Word High Register (RCWHR)”). The attributes should be programmed as follows: alternate configuration space (ACS) should be cleared (0b0), byte enables should be all ones, and continue (CONT) should be set.

After the first 3 bytes, 4 bytes of data should hold the desired value of the reset configuration word. The boot sequencer assumes that a big-endian address is stored in the EEPROM.

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

When the I^2C operates in reset configuration mode, the cyclic redundancy check (CRC) is ignored, as well as any registers following the first two reset configuration words.

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

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

Figure 4-8 shows an example of the EEPROM contents, including the preamble, reset configuration words and additional initialization data, and CRC. In this example, it is assumed that the EEPROM contains information additional to the reset configuration words, which should be loaded in the functional state after the device completes its reset flow.

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

Figure 4-8. EEPROM Contents

4.3.3.2.4 Reset Configuration Load Fail

Failure of reset configuration load by the I^2C boot sequencer can be caused by an incorrect EEPROM data structure or I^2C bus problem. If a reset configuration load failure occurs, due to preamble fail or any other I^2C bus error detection, the device will continuously attempt to reload the hard reset configuration words from the I^2C bus. The device does not negate $\overline{\text{HRESET}}$ and remains in hard reset state until the HRCWs are successfully loaded or the PORESET flow is restarted.

4.3.3.3 Default Reset Configuration Words

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

The reset configuration words are driven internally with the values shown in Table 4-21 and Table 4-22.

NOTE

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

Table 4-21. Hard Coded Reset Configuration Word Low Fields Values

RCWL Bits:	0	1	2–3	4–7	8	9–15	16–17	18–23	24–25	26	27–31
Field:	**LBCM**	**DDRCM**	**Res**	**SPMF**	**Res**	**COREPLL**	**Res**	**Res**	**CEVCOD**	**CEPDF**	**CEPMF**
Meaning: / **CFG_RESET_SOURCE Value**	LBC controller clock: *csb_clk* 0 1:1 1 2:1	DDR controller clock: *csb_clk* 0 1:1 1 2:1	—	*csb_clk*: PCI_CLK ratio SPMF: 1	—	Core clock: *csb_clk* ratio	—	—	QUICC Engine PLL VCO division	QUICC Engine PLL division factor	QUICC Engine Clock: PCI_CLK ratio CEPMF: 1
011	0	1	10	0100	0	0000100	00	000000	10	0	00110
100	0	1	10	0010	0	0000100	00	000000	10	0	00011
101	0	1	10	0100	0	0000101	00	000000	10	0	00110
110	0	1	10	0100	0	0000110	00	000000	10	0	00110
111	0	1	10	0011	0	0000100	00	000000	10	0	00110

Table 4-22 defines the hard-coded reset configuration word high fields values. These values select hard-coded reset configuration words options, as described in Section 4.3.1.1, "Reset Configuration Word Source."

Table 4-22. Hard-Coded Reset Configuration Word High Field Values

Bits	Name	CFG_RESET_SOURCE[0:2] = 011–111	Meaning
0	PCIHOST	0	PCI agent mode
1	Reserved	1	—
2	PCIARB	0	External arbiter is used
3	Reserved	0	—
4	COREDIS	1	e300 core is disabled (boot holdoff)
5	BMS	1	Boot memory space is 0xFF80_0000–0xFFFF_FFFF. MSR[IP] initial value is 0b1
6–7	BOOTSEQ	00	Boot sequencer is disabled.
8	SWEN	0	Software watchdog disabled.
9–11	ROMLOC	000	Boot ROM interface location.
12–15	Reserved	0000	—
16–19	Reserved	1010	—
20–27	Reserved	0000_0000	—
28	TLE	0	Big endian mode
29	LALE	0	Normal timing
30–31	Reserved	00	—

4.3.3.3.1 Examples for Hard-Coded Reset Configuration Words Usage

Examples for various clock modes are listed in Table 4-23.

Table 4-23. Examples For Hard Coded Reset Configuration Words Usage

CFG_RESET_SOURCE[0:2]	011	100	101	110	111
PCI_CLK (MHz)	33	66	33	33	33
csb_clk (MHz)	133	133	133	133	100
ddr_clk (MHz)	266	266	266	266	266
Core Clock (MHz)	266	266	333	400	200
QUICC Engine Clock (MHz)	200	200	200	200	200

4.4 Clocking

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

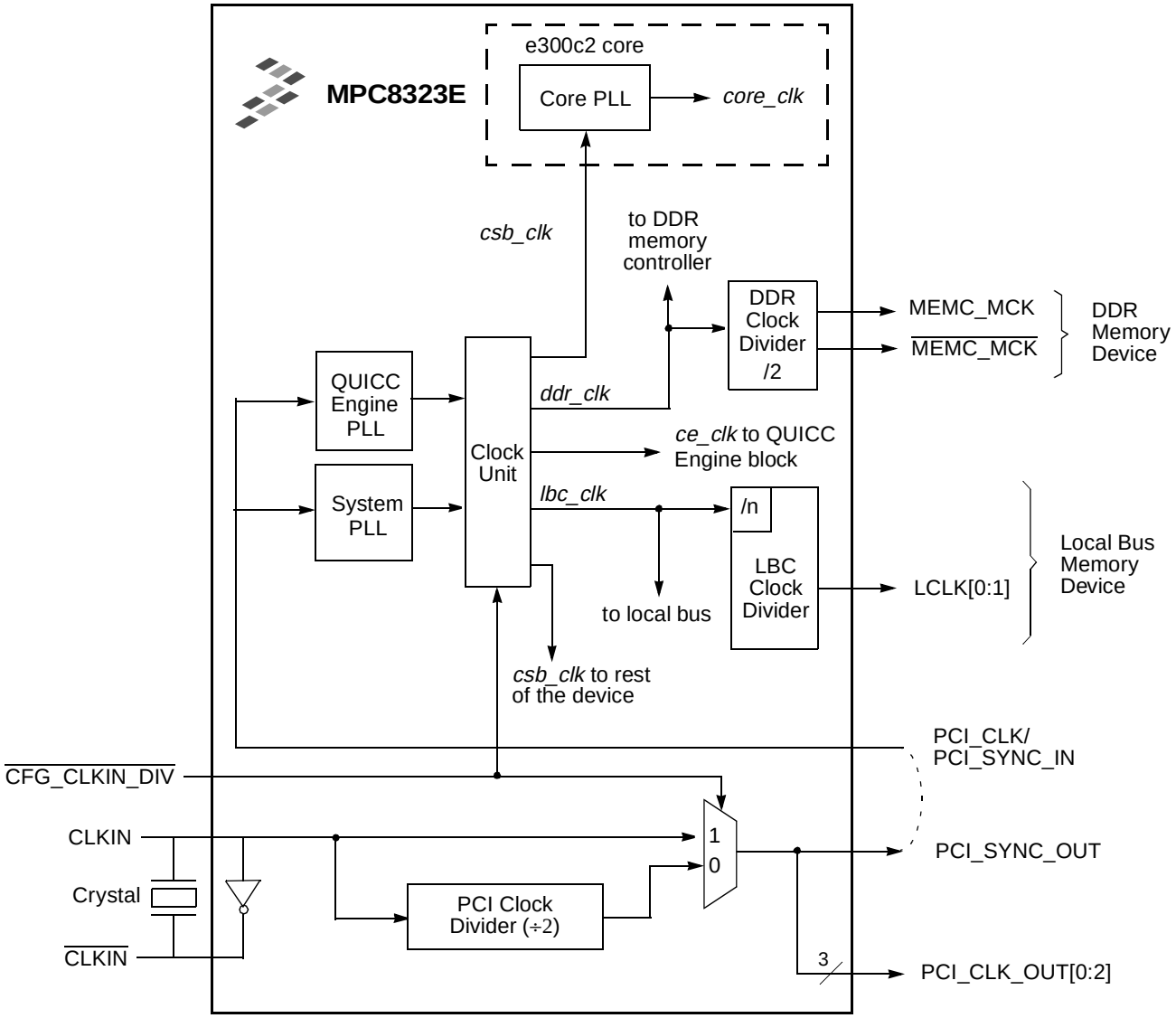


Figure 4-9. Clock Subsystem Block Diagram

The primary clock source for the device can be one of two inputs, CLKIN or PCI_CLK, depending on whether the device is configured in PCI host or PCI agent mode, respectively.

4.4.1 Clocking in PCI Host Mode

When the device is configured as a PCI host device (RCWH[PCIHOST] = 1), CLKIN is the primary input clock. CLKIN feeds the PCI clock divider (÷2) and the PCI_SYNC_OUT and PCI_CLK_OUT multiplexors. The $\overline{\text{CFG\_CLKIN\_DIV}}$ configuration input selects whether CLKIN or CLKIN/2 is driven out on the PCI_SYNC_OUT signal.

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

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

When the device is configured as a PCI host, it provides three clock output signals, PCI_CLK_OUT[0:2], for external PCI agents.

When the device comes out of reset, the PCI clock outputs are disabled and are actively driven to a steady low state. Each of the individual clock outputs can be enabled (enable toggling of the clock) by setting its corresponding OCCR[PCICOE*n*] bit. All output clocks are phase aligned to each other and to PCI_SYNC_OUT.

4.4.2 Clocking In PCI Agent Mode

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

In agent mode, the $\overline{\text{CFG\_CLKIN\_DIV}}$ configuration input can be used to double the internal clock frequencies, if sampled as 0 during PORESET assertion. This feature is useful if a fixed internal frequency is desired regardless of whether the PCI clock is running at 33 or 66 MHz. PCI specifications require that the signal M66EN provides the PCI clock frequency information.

4.4.3 System Clock Domains

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

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

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

$$csb\_clk = [\text{PCI\_SYNC\_IN} \times (1 + \sim\overline{\text{CFG\_CLKIN\_DIV}})] \times \text{SPMF}$$

In PCI host mode, PCI_SYNC_IN × (1 + ~$\overline{\text{CFG\_CLKIN\_DIV}}$) is the CLKIN frequency.

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

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

When CLKIN is the primary input clock,

ce_clk = (primary clock input × CEPMF) ÷ (1 + CEPDF)

When PCI_CLK is the primary input clock,

ce_clk = [primary clock input × CEPMF × (1 + ~CFG_CLKIN_DIV)] ÷ (1 + CEPDF)

See Section 4.3.2.1.2, "QUICC Engine PLL Multiplication Factor," and Section 4.3.2.1, "Reset Configuration Word Low Register (RCWLR)," for more information.

The DDR SDRAM memory controller will operate with a frequency equal to twice the frequency of *csb_clk*. Note that *ddr_clk* is not the external memory bus frequency; *ddr_clk* passes through the DDR clock divider (÷2) to create the differential DDR memory bus clock outputs (MCK and $\overline{\text{MCK}}$). However, the data rate is the same frequency as *ddr_clk.*

The local bus memory controller will operate with a frequency equal to the frequency of *csb_clk*. Note that *lbc_clk* is not the external local bus frequency; *lbc_clk* passes through the LBC clock divider to create the external local bus clock outputs (LSYNC_OUT and LCLK[0:2]). The LBC clock divider ratio is controlled by LCCR[CLKDIV]. See Section 10.1.2.1, "LBC Bus Clock and Clock Ratios," for more information.

In addition, some of the internal units may be required to be shut off or operate at lower frequency than the *csb_clk* frequency. These units have a default clock ratio that can be configured by a memory mapped register after the device comes out of reset. Table 4-24 specifies which units have a configurable clock frequency. Refer to Section 4.5.2.3, "System Clock Control Register (SCCR)."

Table 4-24. Configurable Clock Units

Unit	Default Frequency	Options
Security core, I^2C	*csb_clk*	Off, *csb_clk*/2, *csb_clk*/3
PCI and DMA complex	*csb_clk*	Off, *csb_clk*

NOTE

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

4.5 Memory Map/Register Definitions

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

4.5.1 Reset Configuration Register Descriptions

The reset configuration and status registers are shown in Table 4-25.

Table 4-25. Reset Configuration and Status Registers Memory Map

Address	Register	Access	Reset	Section/Page
0x0_0900	Reset configuration word low register (RCWLR)	R	0x0000_0000	4.5.1.1/4-30
0x0_0904	Reset configuration word high register (RCWHR)	R	0x0000_0000	4.5.1.2/4-30
0x0_0908	Reserved, should be cleared	—	—	—
0x0_090C	Reserved, should be cleared	—	—	—
0x0_0910	Reset status register (RSR)	R/W	0x0000_0000	4.5.1.3/4-31
0x0_0914	Reset mode register (RMR)	R/W	0x0000_0000	4.5.1.4/4-32
0x0_0918	Reset protection register (RPR)	R/W	0x0000_0000	4.5.1.5/4-33
0x0_091C	Reset control register (RCR)	R/W	0x0000_0000	4.5.1.6/4-33
0x0_0920	Reset control enable register (RCER)	R/W	0x0000_0000	4.5.1.7/4-34
0x0_0924–0x0_09FC	Reserved, should be cleared.	—	—	—

4.5.1.1 Reset Configuration Word Low Register (RCWLR)

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

4.5.1.2 Reset Configuration Word High Register (RCWHR)

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

4.5.1.3 Reset Status Register (RSR)

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

Address 0x0_0910 Access: User read/write

	0	2	3									14	15
R	RSTSRC		—										BSF
W													
Reset	n1		0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	22	23	24	26	27	28	29	30	31
R	—		SWSR	SWHR	—		JSRS	—		CSHR	SWRS	BMRS	SRS	HRS
W														
Reset	All zeros													

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

Figure 4-10. Reset Status Register (RSR)

Table 4-26 defines the reset status register bit fields.

Table 4-26. Reset Status Register Field Descriptions

Bits	Name	Description
0–2	RSTSRC	Reset configuration word source. Reflects the value of CFG_RESET_SOURCE input signal during the reset flow. See Section 4.3.1.1, “Reset Configuration Word Source,” on page 4-9. Changing this field has no effect.
3–14	—	Reserved, should be cleared.
15	BSF	Boot sequencer fail. If set, indicates that the I^2C boot sequencer has failed while loading the reset configuration words. Cleared by writing a 1 to it (writing zero has no effect).
16–17	—	Reserved, should be cleared.
18	SWSR	Software soft reset. If set, indicates that a software soft reset has occurred. Cleared by writing a 1 to it (writing zero has no effect).
19	SWHR	Software hard reset. If set, indicates a software hard reset. SWHR is cleared by writing a 1 to it (writing zero has no effect).
20–22	—	Reserved, should be cleared.
23	JSRS	JTAG soft reset status. Set when the JTAG reset request is set and remains set until software clears it. JSRS is cleared by writing a 1 to it (writing zero has no effect). 0 No JTAG reset event. 1 JTAG reset event.
24–26	—	Reserved, should be cleared.
27	CSHR	Check stop reset status. When the core enters a checkstop state and the checkstop reset is enabled by the RMR[CSRE], CSRS is set and it remains set until software clears it. CSRS is cleared by writing a 1 to it (writing zero has no effect). 0 No enabled check stop reset event. 1 Enabled check stop reset event.

Table 4-26. Reset Status Register Field Descriptions (continued)

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

4.5.1.4 Reset Mode Register (RMR)

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

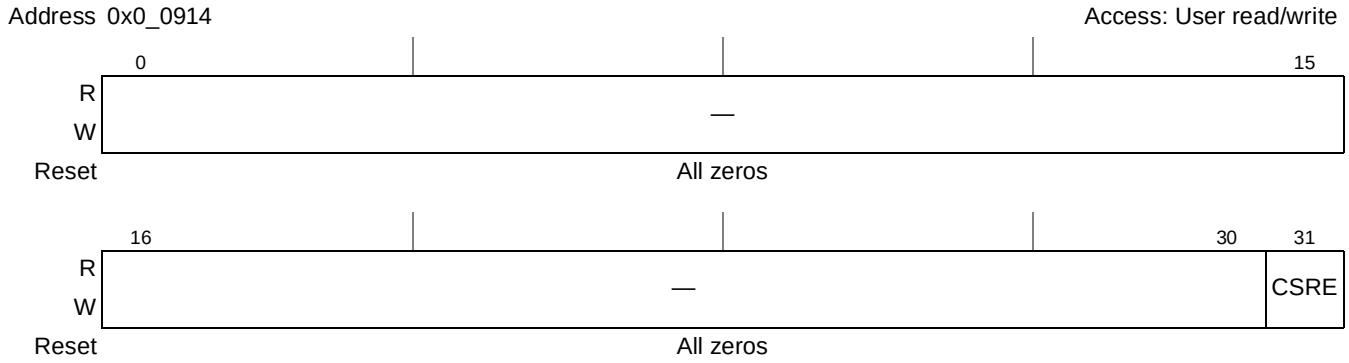


Figure 4-11. Reset Mode Register (RMR)

Table 4-27 describes the RMR fields.

Table 4-27. RMR Field Descriptions

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

4.5.1.5 Reset Protection Register (RPR)

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

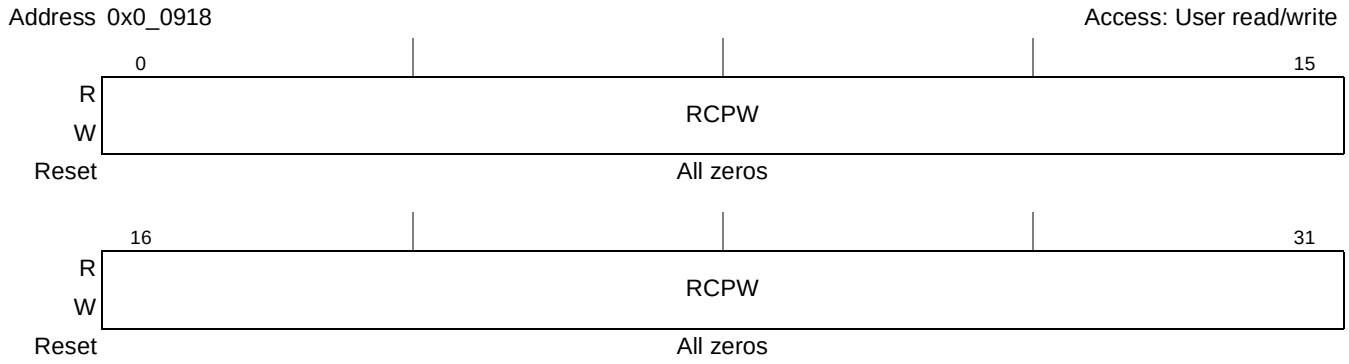


Figure 4-12. Reset Protection Register (RPR)

Table 4-28 defines the bit fields of RPR.

Table 4-28. RPR Bit Descriptions

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

4.5.1.6 Reset Control Register (RCR)

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

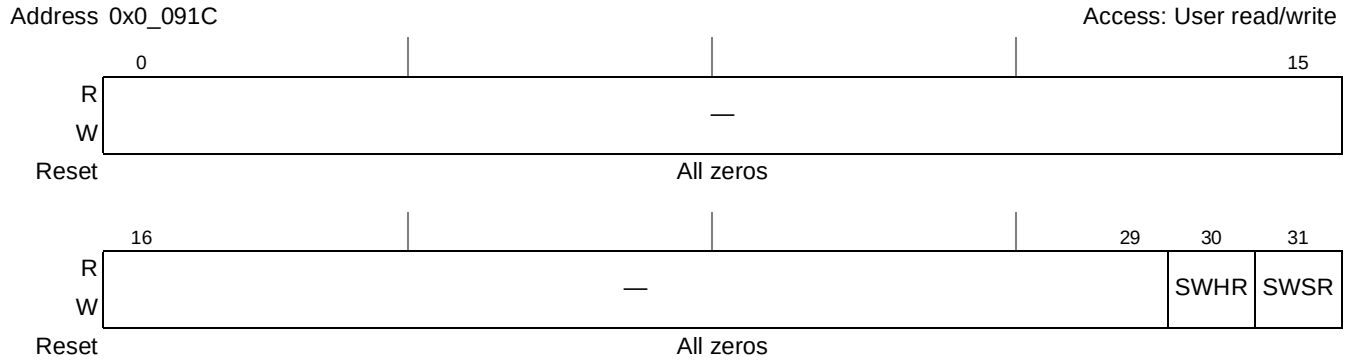


Figure 4-13. Reset Control Register (RCR)

Table 4-29 defines the bit fields of RCR.

Table 4-29. RCR Bit Settings

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

4.5.1.7 Reset Control Enable Register (RCER)

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

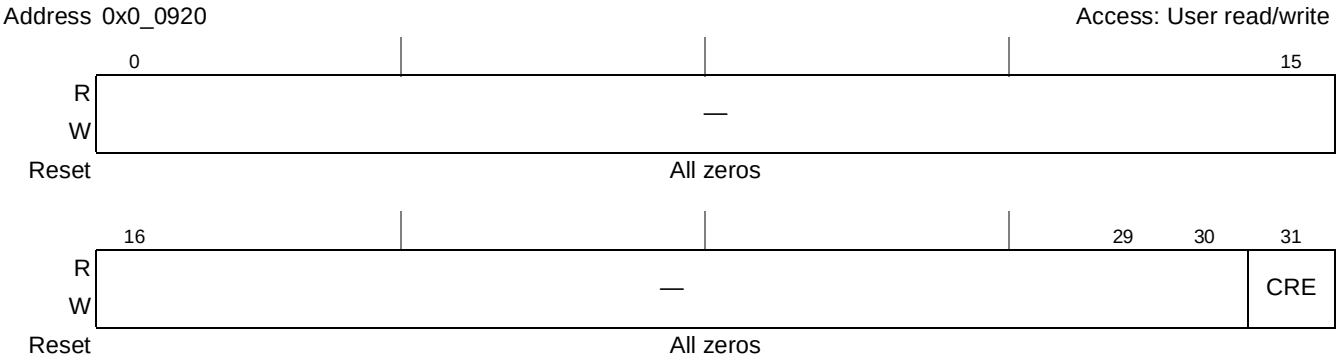


Figure 4-14. Reset Control Enable Register (RCER)

Table 4-30 defines the bit fields of RCER.

Table 4-30. RCER Bit Settings

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

4.5.2 Clock Configuration Registers

The clock configuration and status registers are shown in Table 4-31.

Table 4-31. Clock Configuration Registers Memory Map

Address	Register	Access	Reset	Section/Page
0x0_0A00	System PLL mode register (SPMR)	R	0x*nnnn_nnnn*	4.5.2.1/4-35
0x0_0A04	Output clock control register (OCCR)	R/W	0x0000_0000	4.5.2.2/4-36

Table 4-31. Clock Configuration Registers Memory Map (continued)

Address	Register	Access	Reset	Section/Page
0x0_0A08	System clock control register (SCCR)	R/W	0xFDFF_FFFF	4.5.2.3/4-37
0x0_0A0C–0x0_0AFC	Reserved, should be cleared	—	—	—

4.5.2.1 System PLL Mode Register (SPMR)

SPMR is shown in Figure 4-15, gets its values according to the $\overline{\text{CFG\_CLKIN\_DIV}}$ reset configuration input signal and the reset configuration word low loaded during the reset flow. Note that this register is updated only during a power-on reset sequence and not by a hard reset sequence. It may hold values different than those in the RCWLR after a a hard reset sequence.

Address 0x0_0A00 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LBCM	DDRCM	—		SPMF				CKID	COREPLL						
W																
Reset[1]	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	—								CEVCOD		CEPDF	CEPMF				
W																
Reset	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

Figure 4-15. System PLL Mode Register

[1] See Table 4-32 for reset values.

Table 4-32 defines the system PLL mode register bit fields.

Table 4-32. System PLL Mode Register Bit Settings

Bits	Name	Meaning	Description
0	LBCM	Local bus memory controller clock mode.	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR)”
1	DDRCM	DDR SDRAM memory controller clock mode.	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR)”
2–3	—	Reserved, should be cleared.	—
4–7	SPMF	System PLL multiplication factor	Section 4.3.2.1.1, “System PLL Configuration”
8	CKID	CLKIN division factor. Reflects the value of $\overline{\text{CFG\_CLKIN\_DIV}}$ input signal during the reset flow.	Section 4.3.1.2, “CLKIN Division”
9–15	COREPLL	Core PLL configuration.	See the hardware specifications for this device
16–23	—	Reserved, should be cleared.	—
24–25	CEVCOD	QUICC Engine PLL VCO division	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR).”

Table 4-32. System PLL Mode Register Bit Settings (continued)

Bits	Name	Meaning	Description
26	CEPDF	QUICC Engine PLL division factor	Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR)”
27–31	CEPMF	QUICC Engine PLL multiplication factor	Section 4.3.2.1.2, “QUICC Engine PLL Multiplication Factor”

4.5.2.2 Output Clock Control Register (OCCR)

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

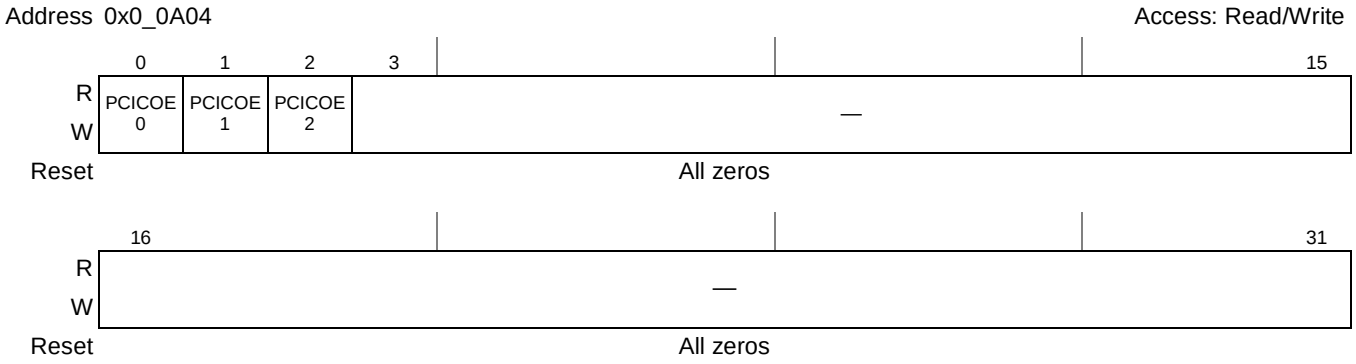


Figure 4-16. Output Clock Control Register (OCCR)

Table 4-33 defines the bit fields of OCCR.

Table 4-33. OCCR Bit Settings

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

4.5.2.3 System Clock Control Register (SCCR)

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

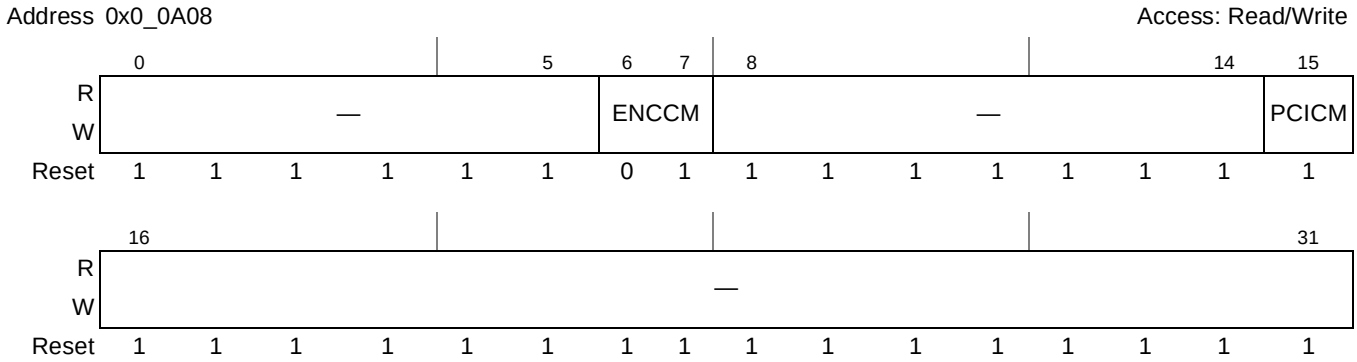


Figure 4-17. System Clock Control Register (SCCR)

Table 4-34 defines the bit fields of SCCR.

Table 4-34. SCCR Bit Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6–7	ENCCM	Encryption core and I^2C clock mode. 00 Encryption core clock is disabled. 01 Encryption core clock/*csb_clk* ratio is 1:1. 10 Encryption core clock/*csb_clk* ratio is 1:2 (*csb_clk* has higher frequency than the encryption core). 11 Encryption core clock/*csb_clk* ratio is 1:3 (*csb_clk* has higher frequency than the encryption core).
8–14	—	Reserved, should be cleared.
15	PCICM	PCI clock mode. Define the clock mode for all of the PCI complex - PCI and DMA. 0 PCI complex clocks are disabled. 1 PCI complex clocks are enabled.
16–31	—	Reserved

4.5.3 Clock Control DDR Register

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

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

Table 4-35. Clock Control DDR Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00–0x0F	Reserved, should be cleared.	R	0x0000_0000	—
0x10	MCK enable register (MCKENR)	R/W	0xC000_0000	4.5.3.1/4-38
0x14–0xFF	Reserved, should be cleared.	—	—	—

4.5.3.1 MCK Enable Register (MCKENR)

MCKENR, shown in Figure 4-18, enables or disables the DDR clock outputs.

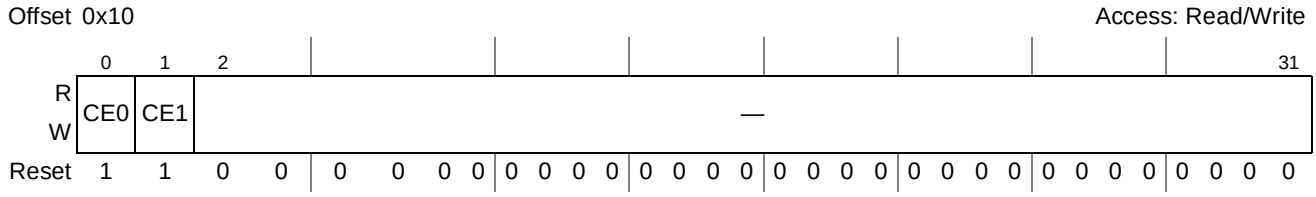


Figure 4-18. MCK Enable Register (MCKENR)

Table 4-36 describes the MCKENR fields.

Table 4-36. MCKENR Field Descriptions

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

4.5.4 Clock Control LBC Registers

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

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

Table 4-37. Programmable Clock Control LBC register map

Offset	Register	Access	Reset Value	Section/ Page
0x00–0x0F	Reserved, should be cleared.	R	0x0000_0000	—
0x10	LCLK enable register (LCLKENR) **Note:** Offset 0x10 is an offset from the clock control LBC base offset 0x0_1100	R/W	0xC000_0000	4.5.4.1/4-39
0x14–0xFF	Reserved, should be cleared.	—	—	—

4.5.4.1 LCLK Enable Register (LCLKENR)

LCLKENR, shown in Figure 4-19, enables or disables the LBC clock outputs.

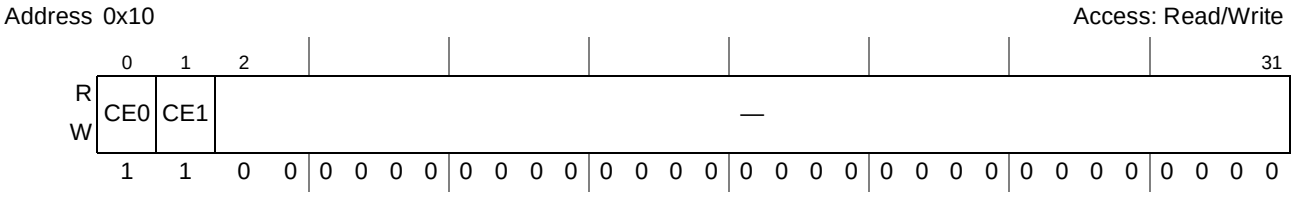


Figure 4-19. LCLKENR Clock Register (LCLKENR)

Table 4-38 describes LCLKENR fields.

Table 4-38. LCLKENR Field Descriptions

Bits	Name	Description
0	CE0	Enable/Disable LCLK[0] clock output signals 0 Disable LCLK[0] 1 Enable LCLK[0]
1	CE1	Enable/Disable LCLK[1] clock output signals 0 Disable LCLK[1] 1 Enable LCLK[1]
2–31	—	Reserved, should be cleared.

Chapter 5
System Configuration

5.1 Introduction

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

- Section 5.2, “Local Memory Map Overview and Example”
- Section 5.3, “System Configuration”
- Section 5.4, “Software Watchdog Timer (WDT)”
- Section 5.5, “Real Time Clock Module (RTC)”
- Section 5.6, “Periodic Interval Timer (PIT)”
- Section 5.7, “General-Purpose Timers (GTMs)”
- Section 5.8, “Power Management Control (PMC)”

5.2 Local Memory Map Overview and Example

The device provides a flexible local memory map. The local memory map refers to the 32-bit address space seen by the processor as it accesses memory and I/O space. Internal DMA engines also see this same local memory map. All memory accessed by the DDR SDRAM 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 nine local access windows. Each of these windows maps a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. The DSP subsystem is not operational in the MSC7104. Note that the local access windows do not perform any address translation. The size of each window can be configured from 4 Kbytes to 2 Gbytes. Each local access window is assigned to a specific target interface as specified in Table 5-1.

Table 5-1. Local Access Windows Target Interface

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

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

Window Number	Target Interface	Comments
7	DDR SDRAM	—
8	DDR SDRAM	—

Figure 5-1 shows an example memory map.

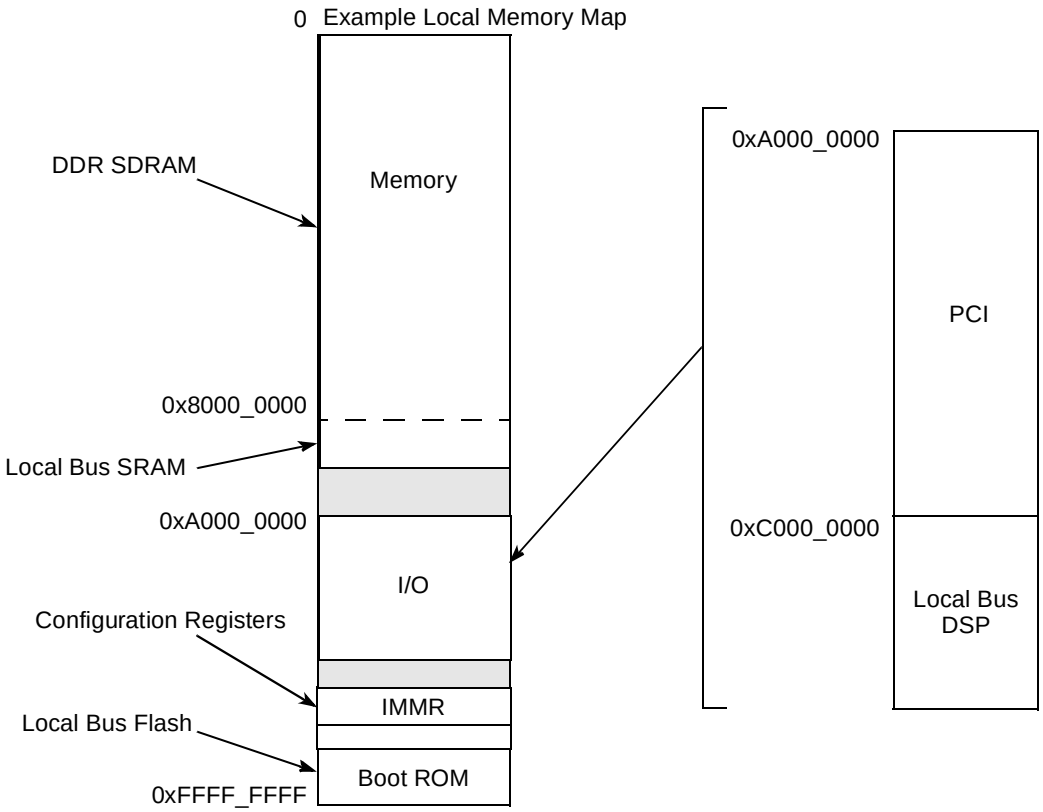


Figure 5-1. Local Memory Map Example

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

Table 5-2. Local Access Windows Example

Window	Base Address	Size	Target Interface
7	0x0000_0000	2 Gbytes	DDR SDRAM
2	0x8000_0000	1 Mbyte	Local bus
5	0xA000_0000	512 Mbytes	PCI
3	0xC000_0000	256 Mbytes	Local bus
0	0xFF40_0000	2 Mbyte	Configuration registers (IMMR)
1	0xFF80_0000	8 Mbytes	Local bus boot ROM Flash
4, 6, 8	Unused		

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

5.2.1 Address Translation and Mapping

In addition to any address translation performed by the e300c2 core MMU, three distinct types of translation and mapping operations are performed on transactions at the integrated device level. These are as follows:

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

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

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

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

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

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

Table 5-3. Format of Window Definitions

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

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

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

5.2.2 Window into Configuration Space

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

NOTE

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

5.2.3 Local Access Windows

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

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

The local access window registers exist as part of the local access block in the system configuration registers. See Section 5.3.2, "System Configuration Registers." A detailed description of the local access window registers is given in the following sections. Note that the minimum size of a window is 4 Kbytes, so the low order 12 bits of the base address cannot be specified.

5.2.3.1 Local Access Register Memory Map

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

Table 5-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	Internal memory map base address register (IMMRBAR)	R/W	0xFF40_0000	5.2.4.1/5-6
0x0_0004	Reserved	—	—	—
0x0_0008	Alternate configuration base address register (ALTCBAR)	R/W	0x0000_0000	5.2.4.2/5-7
0x0_000C–0x0_001C	Reserved	—	—	—
0x0_0020	LBC local access window 0 base address register (LBLAWBAR0)	R/W	0x0000_0000[1]	5.2.4.3/5-8

Table 5-4. Local Access Register Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0024	LBC local access window 0 attribute register (LBLAWAR0)	R/W	0x0000_0000[2]	5.2.4.4/5-9
0x0_0028	LBC local access window 1 base address register (LBLAWBAR1)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_002C	LBC local access window 1 attribute register (LBLAWAR1)	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0030	LBC local access window 2 base address register (LBLAWBAR2)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_0034	LBC local access window 2 attribute register (LBLAWAR2)	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0038	LBC local access window 3 base address register (LBLAWBAR3)	R/W	0x0000_0000	5.2.4.3/5-8
0x0_003C	LBC local access window 3 attribute register (LBLAWAR3)	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0040–0x0_005C	Reserved	—	—	—
0x0_0060	PCI local access window 0 base address register (PCILAWBAR0)	R/W	0x0000_0000[3]	5.2.4.5/5-10
0x0_0064	PCI local access window 0 attribute register (PCILAWAR0)	R/W	0x0000_0000[4]	5.2.4.6/5-11
0x0_0068	PCI local access window 1 base address register (PCILAWBAR1)	R/W	0x0000_0000[5]	5.2.4.5/5-10
0x0_006C	PCI local access window 1 attribute register (PCILAWAR1)	R/W	0x0000_0000	5.2.4.6/5-11
0x0_0070–0x0_009C	Reserved	—	—	—
0x0_00A0	DDR local access window 0 base address register (DDRLAWBAR0)	R/W	0x0000_0000[6]	5.2.4.7/5-12
0x0_00A4	DDR local access window 0 attribute register (DDRLAWAR0)	R/W	0x0000_0000[7]	5.2.4.8/5-13
0x0_00A8	DDR local access window 1 base address register (DDRLAWBAR1)	R/W	0x0000_0000	5.2.4.7/5-12
0x0_00AC	DDR local access window 1 attribute register (DDRLAWAR1)	R/W	0x0000_0000	5.2.4.8/5-13
0x0_00B0–0x0_00FC	Reserved	—	—	—

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

5.2.4 Local Access Register Descriptions

5.2.4.1 Internal Memory Map Registers Base Address Register (IMMRBAR)

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

5.2.4.1.1 Updating IMMRBAR

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

- IMMRBAR should be updated during initial configuration of the device when only one host or controller has access to the device as follows:
  - If an external host on PCI is configuring the device, it should set IMMRBAR to the desired final location before the e300c2 core is released to boot.
  - If the core is initializing the device, it should set IMMRBAR to the desired final location before enabling other I/O devices to access the device.
- When the e300 core is writing to IMMRBAR, it should use the following sequence:
  - Read the current value of IMMRBAR using a load word instruction followed by an **isync**. This forces all accesses to configuration space to complete.
  - Write the new value to IMMRBAR.
  - Perform a load of an address that does not access configuration space or the on-chip SRAM, but has an address mapping already in effect (for example, boot ROM). Follow this load with an **isync**.
  - Read the contents of IMMRBAR from its new location, followed by another **isync**.

The IMMRBAR is shown in Figure 5-2.

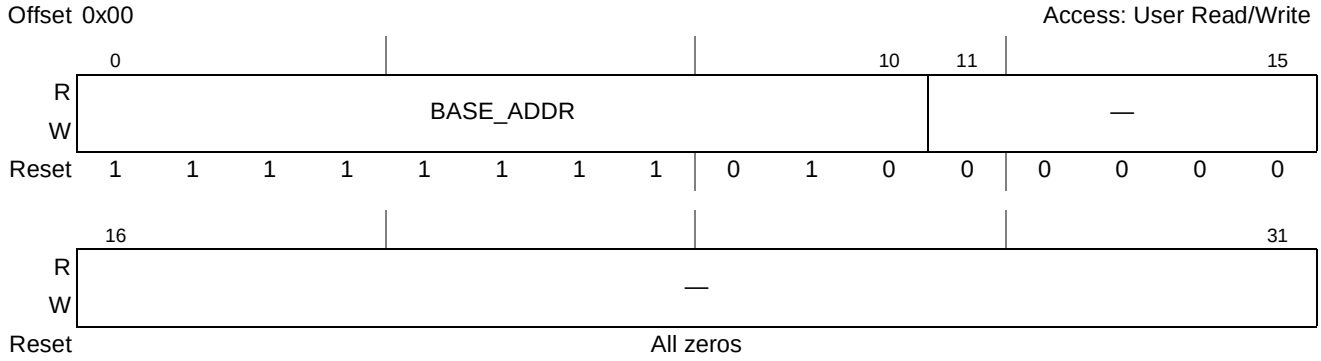


Figure 5-2. Internal Memory Map Registers' Base Address Register (IMMRBAR)

Table 5-5 defines the bit fields of IMMRBAR.

Table 5-5. IMMRBAR Bit Settings

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

5.2.4.2 Alternate Configuration Base Address Register (ALTCBAR)

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

NOTE

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

The alternate configuration base address register is shown in Figure 5-3.

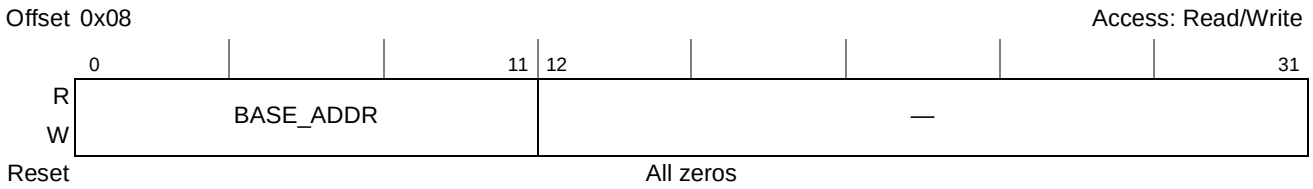


Figure 5-3. Alternate Configuration Base Address Register (ALTCBAR)

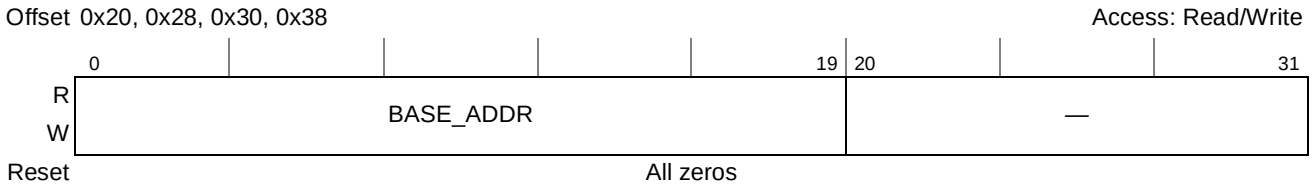
Table 5-6 defines the bit fields of ALTCBAR.

Table 5-6. ALTCBAR Bit Settings

Bits	Name	Description
0–11	BASE_ADDR	Identifies the12 most-significant address bits of an alternate base address used for boot sequencer configuration accesses.
12–31	—	Reserved. Write has no effect, read returns 0.

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

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



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

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

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

Table 5-7. LBLAWBAR0–LBLAWBAR3 Bit Settings

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

5.2.4.3.1 LBLAWBAR0[BASE_ADDR] Reset Value

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

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

Table 5-8. LBLAWBAR0[BASE_ADDR] Reset Value

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

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

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

Offset 0x24, 0x2C, 0x34, 0x3C — Access: Read/Write

	0	1																				25	26						31
R	EN	—																					SIZE						
W																													
Reset	0[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[2]

1 The LBLAWAR0[EN] reset value depends on the reset configuration word high values. See Section 5.2.4.4.1, "LBLAWAR0[EN] and LBLAWAR0[SIZE] Reset Value," for a detailed description.

2 The LBLAWAR0[SIZE] reset value is always 0b010110, meaning an 8-Mbyte local access window. See Section 5.2.4.4.1, "LBLAWAR0[EN] and LBLAWAR0[SIZE] Reset Value," for a detailed description.

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

Table 5-9 defines the bit fields of LBLAWAR0–LBLAWAR3.

Table 5-9. LBLAWAR0–LBLAWAR3 Bit Settings

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

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

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

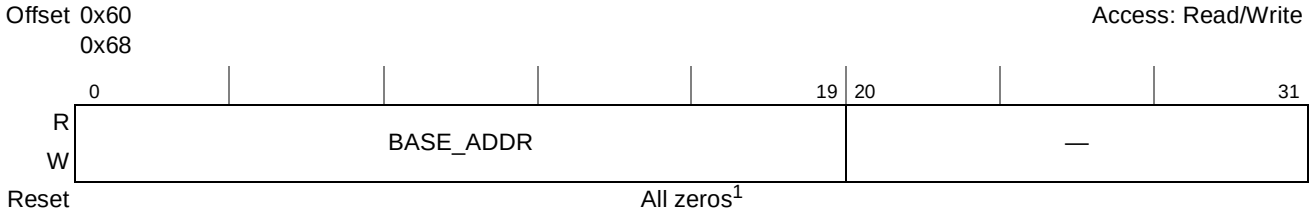
Table 5-10 defines the reset value for LBLAWAR0[EN].

Table 5-10. LBLAWAR0[EN] Reset Value

RCWHR[ROMLOC]	LBLAWAR0[EN] Reset Value	Description
000–100	0	e300c2 core boot not performed from a local bus device.
101–110	1	e300c2 core boot performed from a local bus device. Local bus 8-Mbyte ($2^{(22+1)}$) local access window is enabled.
111	0	Reserved

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

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



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

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

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

Table 5-11. PCILAWBAR0–PCILAWBAR1 Bit Settings

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

5.2.4.5.1 PCILAWBAR0[BASE_ADDR] Reset Value

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

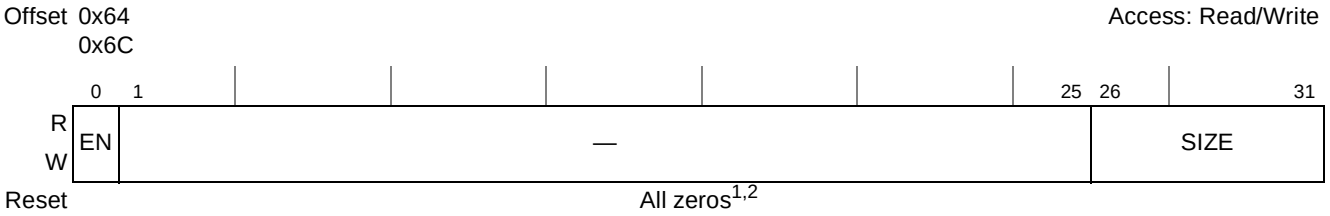
Table 5-12 defines the reset value of PCILAWBAR0[BASE_ADDR].

Table 5-12. PCILAWBAR0[BASE_ADDR] Reset Value

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

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

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



1 The reset value of PCILAWAR0[EN] depends on the reset configuration word high values. See Section 5.2.4.6.1, "PCILAWAR0[EN] and PCILAWAR0[SIZE] Reset Value," for a detailed description.

2 The reset value of PCILAWAR0[SIZE] is always 0b010110, meaning an 8-Mbyte local access window. See Section 5.2.4.6.1, "PCILAWAR0[EN] and PCILAWAR0[SIZE] Reset Value," for a detailed description

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

Table 5-13 defines the bit fields of PCILAWAR0–PCILAWAR1.

Table 5-13. PCILAWAR0–PCILAWAR1 Bit Settings

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

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

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

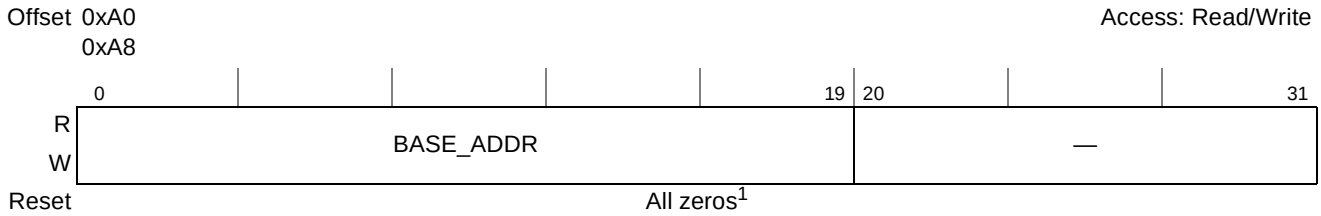
Table 5-14 defines the reset value of PCILAWAR0[EN].

Table 5-14. PCILAWAR0[EN] Reset Value

RCWHR[ROMLOC]	PCILAWR0[EN] Reset Value	Description
000, 010–110	0	e300c2 core boot not performed from a PCI device.
111	0	Reserved

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

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



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

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

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

Table 5-15. DDRLAWBAR0–DDRLAWBAR1 Bit Settings

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

5.2.4.7.1 DDRLAWBAR0[BASE_ADDR] Reset Value

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

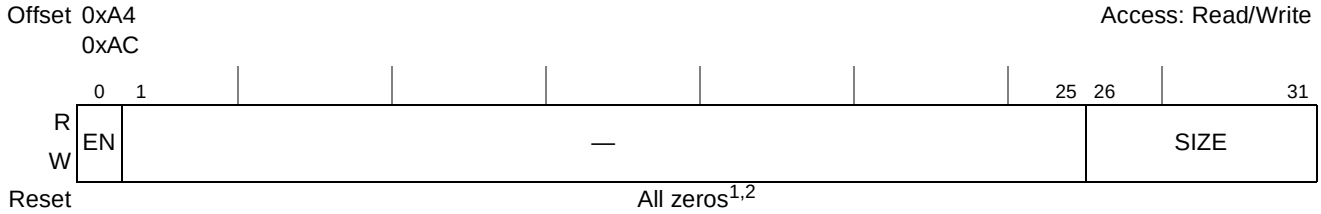
Table 5-16 defines the reset value DDRLAWBAR0.

Table 5-16. DDRLAWBAR0[BASE_ADDR] Reset Value

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

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

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



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

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

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

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

Table 5-17. DDRLAWAR0–DDRLAWAR1 Bit Settings

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

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

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

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

Table 5-18. DDRLAWAR0[EN] Reset Value

RCWHR[ROMLOC]	DDRLAWAR0[EN] Reset Value	Description
000	1	e300c2 core boot performed from a DDR SDRAM device. DDR 8-Mbyte ($2^{(22+1)}$) local access window is enabled.
Else	0	e300c2 core boot not performed from a DDR SDRAM device.
111	0	Reserved

5.2.5 Precedence of Local Access Windows

If two local access windows overlap, the lower numbered window takes precedence (see Table 5-1 for window numbers). For instance, if two windows are set up as shown in Table 5-19, local access window 1 governs the mapping of the 1-Mbyte region from 0x7FF0_0000 to 0x7FFF_FFF, even though the window described in local access window 7 also encompasses that memory region.

Table 5-19. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
1	0x7FF0_0000	1 Mbyte	Local bus
7	0x0000_0000	2 Gbytes	DDR SDRAM

5.2.6 Configuring Local Access Windows

After a local access window is enabled, it should not be modified while any device in the system may be using the window. Accordingly, a new window should not be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last local access window configuration register before enabling any other devices to use the window. For instance, if local bus local access windows 1–3 are being configured in order during the initialization process, the last write (to LBLAWAR3) should be followed by a read of LBLAWAR3 before any devices try to use any of these windows. If the configuration is being done by the local e300c2 core, the read of LBLAWAR3 should be followed by an **isync** instruction.

5.2.7 Distinguishing Local Access Windows from Other Mapping Functions

It is important to distinguish between the mapping function performed by the local access windows and the additional mapping functions that happen at the target interface. The local access windows define how a transaction is routed through the device internal interconnects from the transaction's source to its target. Once the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR SDRAM controller has chip select registers that map a memory request to a particular external device. The local bus controller has base registers that perform a similar function. The PCI interface has outbound address translation units that map the local address into an external address space.

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

5.2.8 Outbound Address Translation and Mapping Windows

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

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

5.2.9 Inbound Address Translation and Mapping Windows

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

5.2.9.1 PCI Inbound Windows

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

5.2.10 Internal Memory Map

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

NOTE

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

5.2.11 Accessing Internal Memory from External Masters

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

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

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

5.3 System Configuration

The following sections describe some general information and configuration options that affect system behavior and performance.

5.3.1 System Configuration Register Memory Map

Table 5-20 shows the memory map for the system configuration registers.

Table 5-20. System Configuration Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00100	System general purpose register low (SGPRL)	R/W	0x0000_0000	5.3.2.1/5-17
0x00104	System general purpose register high (SGPRH)	R/W	0x0000_0000	5.3.2.2/5-17
0x00108	System part and revision ID register (SPRIDR)	R	0x8060_0010	5.3.2.3/5-18
0x0010C	Reserved	—	—	—
0x00110	System priority configuration register (SPCR)	R/W	0x0000_0000	5.3.2.4/5-19
0x00114	System I/O configuration register low (SICRL)	R/W	0x0000_0000[1]	5.3.2.5/5-21
0x00118–0x00124	Reserved	—	—	—
0x00128–0x001FC	Reserved	—	—	—

[1] Depends on the reset configuration word high configuration values.

5.3.2 System Configuration Registers

5.3.2.1 System General Purpose Register Low (SGPRL)

The system general purpose register low (SGPRL), shown in Figure 5-10, can be used by software for any purpose. The values set in this register have no effect on the internal hardware.

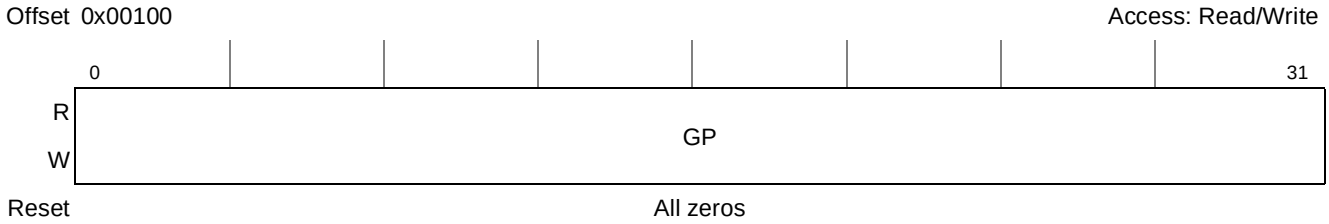


Figure 5-10. System General Purpose Register Low (SGPRL)

Table 5-21 defines the bit fields of SGPRL.

Table 5-21. SGPRL Bit Settings

Bits	Name	Description
0–31	GP	General purpose

5.3.2.2 System General Purpose Register High (SGPRH)

The system general purpose register high (SGPRH), shown in Figure 5-11, can be used by software for any purpose. The values set in this register have no effect on the internal hardware.

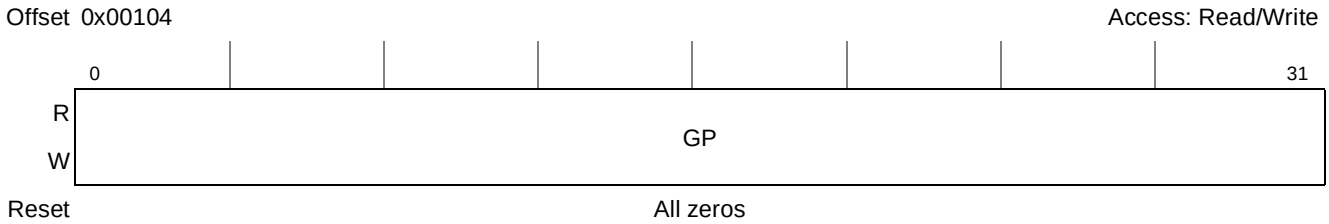


Figure 5-11. System General Purpose Register High (SGPRH)

Table 5-22 defines the bit fields of SGPRH.

Table 5-22. SGPRH Bit Settings

Bits	Name	Description
0–31	GP	General purpose

5.3.2.3 System Part and Revision ID Register (SPRIDR)

SPRIDR, shown in Figure 5-12, provides information about the device and revision numbers.

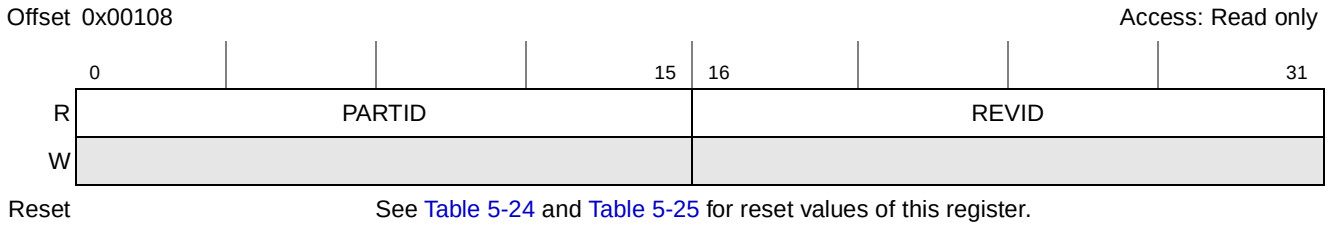


Figure 5-12. System Part and Revision ID Register (SPRIDR)

Table 5-23 defines the bit fields of SPRIDR.

Table 5-23. SPRIDR Bit Settings

Bits	Name	Description
0–15	PARTID	Part identification. This read-only field is mask-programmed with a code corresponding to the device number. It is intended to help factory test and user code that is sensitive to device changes. The device number changes according to manufacturing considerations. See Table 5-24 for values of this field.
16–31	REVID	Revision identification. This read-only field is mask-programmed with a code corresponding to the revision number of the part defined in PARTID field. It is intended to help factory test and user code that is sensitive to device changes. The mask number is programmed in a commonly changed layer, and changes with each mask set change. See Table 5-25 for values of this field.

5.3.2.3.1 SPRIDR[PARTID] Coding

Table 5-24 defines the reset values of SPRIDR[PARTID].

Table 5-24. PARTID Coding [1]

PARTID	Device Name	Package Type
0x8062	MPC8323E	PBGA
0x8063	MPC8323	PBGA
0x8066	MPC8321E	PBGA
0x8067	MPC8321	PBGA

Note:

1 The MSC7121 is functionally identical to the MSC7120. The MSC7104, except for not supporting the use of the integrated StarCore™ DSP, is functionally identical to the MSC7120. All three parts have the same PartID.

With the DSP clock on, HSTAT[RESET] bit remaining cleared after writing a 1 to VEC[S_RESET] indicates that the part is an MSC7104. Once the device has been identified as an MSC7104, the DSP clock should be disabled to save power. DSP clock disabling is done by clearing bit 19 (DSP clock) in the System Clock Control Register (SCCR).

With the DSP clock on, HSTAT[RESET] bit getting set after writing a 1 to VEC[S_RESET] indicates that the part is an MSC7120/MSC7121.

Table 5-25 defines the reset values of SPRIDR[REVID].

Table 5-25. REVID Coding [1]

REVID	Device Revision
0x0010	1.0
0x0011	1.1

1 Revision 1.0 and 2.0 only applies to the MSC7120.

5.3.2.4 System Priority and Configuration Register (SPCR)

The system priority and configuration register (SPCR), shown in Figure 5-13, controls the priority of requests for transactions on the internal system bus. This priority is considered by the system arbiter whenever an internal unit requests mastership of the coherent system bus (CSB). The SPCR also includes some other control functions.

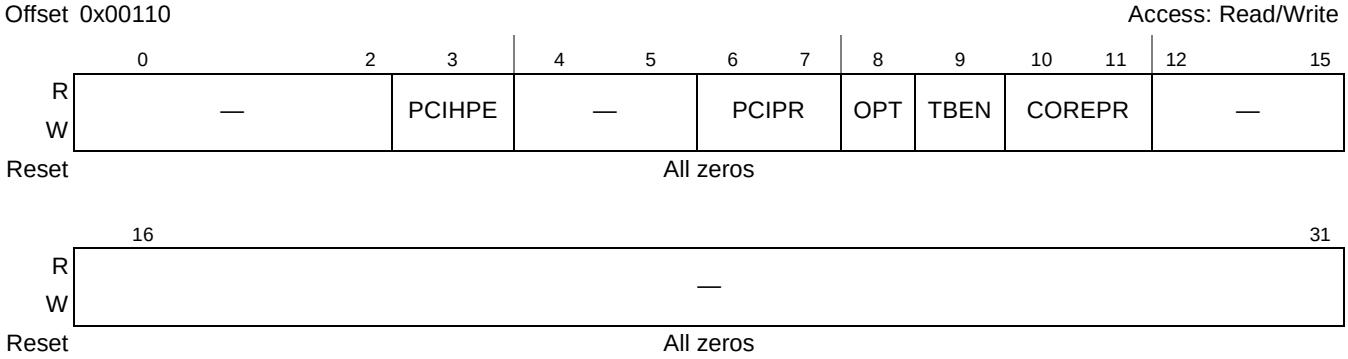


Figure 5-13. System Priority Configuration Register (SPCR)

Table 5-26 defines the bit fields of SPCR.

Table 5-26. SPCR Bit Settings

Bits	Name	Description
0–2	—	Reserved. Should be cleared.
3	PCIHPE	PCI highest priority enable. If this bit is set, the PCI bridge is permitted to request the coherent system bus (CSB) with highest priority, regardless of SPCR[PCIPR] value, when it needs to complete a posted write transaction from an external PCI master. To follow PCI ordering rules specifications, the PCI bridge must flush any outstanding write transactions before it can start a new read transaction. Setting this bit allows faster flushing of the outstanding write transactions coming from the PCI bus onto the CSB and to the device targets, such as DDR SDRAM and local bus memories.
4–5	—	Reserved. Should be cleared.
6–7	PCIPR	PCI bridge CSB request priority. The level of priority can be chosen from 4 possible levels. 00 Level 0 (lowest priority) 01 Level 1 10 Level 2 11 Level 3 (highest priority) **Note:** DMA has the same priority as PCI.

Table 5-26. SPCR Bit Settings (continued)

Bits	Name	Description
8	OPT	Optimize. Setting this bit may enhance the performance of transactions issued to the internal coherent system bus (CSB) by the security engine (SEC) and the QUICC Engine block. Performance is enhanced by reading more bytes on the bus than actually needed by the master in the case that this is more efficient. The user may set this bit only if it is known that QUICC Engine SEC transactions sent to the internal CSB are not accessing devices in which speculative reads may change the state of the device (for example, FIFOs in which reading a byte may advance some internal counter). 0 No performance enhancement. 1 Performance enhancement by speculative reading is enabled.
9	TBEN	e300c2 core time base unit enable 0 Time base unit is disabled. 1 Time base unit is enabled.
10–11	COREPR	e300c2 core CSB request priority. The priority level for the core in accessing the CSB can be chosen from 4 possible levels. 00 Level 0 (lowest priority) 01 Level 1 10 Level 2 11 Level 3 (highest priority)
12–31	—	Reserved. Should be cleared.

5.3.2.5 System I/O Configuration Register Low (SICRL)

The system I/O configuration register low (SICRL) controls the multiplexing of some of the device I/O pins. Each bit or set of bits in the SICRL selects which function is used by a certain group of the device pins.

Figure 5-14 shows SICRL.

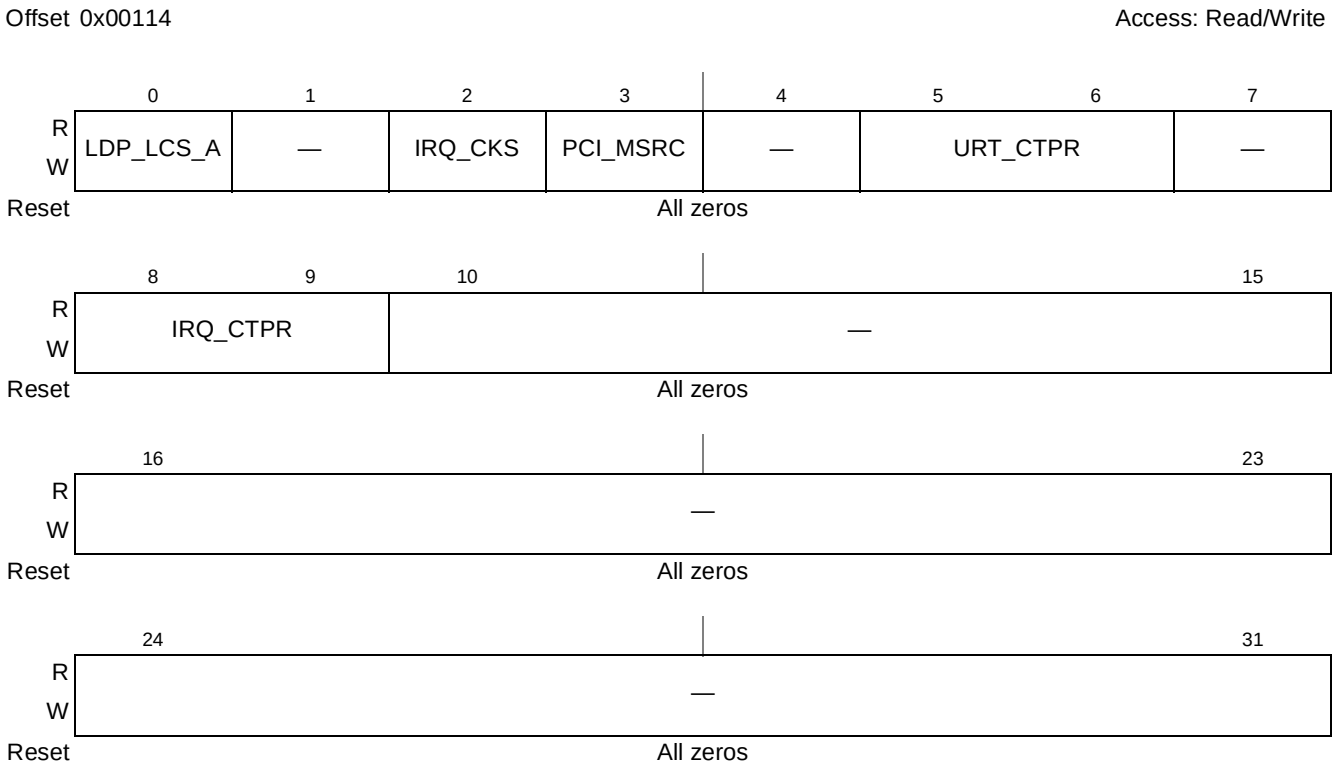


Figure 5-14. System I/O Configuration Register Low (SICRL)

Table 5-27 defines the bit fields of SICRL. Each Pin Function column lists the name of the multi-function pin used in this option. Some groups have only two options (shown as Pin Function 0 and Pin Function 1) and therefore, only one control bit. In this case they can only have a value of 0b0 or 0b1. Other groups may have four options (shown as Pin Function 0, Pin Function 1, Pin Function 2, and Pin Function 3) and therefore, two control bits. In this case they can have a value of 0b00, 0b01, 0b10, or 0b11. Use the notations '0b*N*' or '0b*NN*' according to whether a group has one or two control bits, respectively.

Table 5-27. SICRL Bit Settings

SICRL[Bits] Value:		0b0/0b00	0b1/0b01	0b10	0b11
Bits	**Group**	**Pin Function 0**	**Pin Function 1**	**Pin Function 2**	**Pin Function 3**
0	LDP_LCS_A	IIC_SDA	$\overline{\text{CKSTOP\_OUT}}$	N/A	N/A
		IIC_SCL	$\overline{\text{CKSTOP\_IN}}$	N/A	N/A
		N/A	N/A	N/A	N/A
		N/A	N/A	N/A	N/A

Table 5-27. SICRL Bit Settings (continued)

SICRL[Bits] Value:		0b0/0b00	0b1/0b01	0b10	0b11
Bits	**Group**	**Pin Function 0**	**Pin Function 1**	**Pin Function 2**	**Pin Function 3**
1	Reserved	—	—	—	—
2	IRQ_CKS	IRQ6	CKSTOP_OUT	N/A	N/A
		IRQ7	CKSTOP_IN	N/A	N/A
3	PCI_MSRC	PCI_AD0	MSRCID0	N/A	N/A
		PCI_AD1	MSRCID1	N/A	N/A
		PCI_AD2	MSRCID2	N/A	N/A
		PCI_AD3	MSRCID3	N/A	N/A
		PCI_AD4	MSRCID4	N/A	N/A
		PCI_AD5	MDVAL	N/A	N/A
4	Reserved	—	—	—	—
5–6	URT_CTPR	UART1_SOUT	MSRCID0	—	LSRCID0
		UART1_SIN	MSRCID1	—	LSRCID1
		UART1_CTS	MSRCID2	—	LSRCID2
		UART1_RTS	MSRCID3	—	LSRCID3
		UART2_SOUT	MSRCID4	—	LSRCID4
		UART2_SIN	MDVAL	—	LDVAL
		UART2_CTS	N/A	—	N/A
		UART2_RTS	N/A	—	N/A
		N/A	N/A	N/A	N/A
		N/A	N/A	N/A	N/A
7	Reserved	—	—	—	—
8–9	IRQ_CTPR	$\overline{\text{IRQ0}}$	N/A	—	N/A
		$\overline{\text{IRQ1}}$	N/A	—	N/A
10–31	Reserved	—	—	—	—

5.3.2.6 Debug Configuration

Debug information may be driven on the device pins. This information can identify the internal source of a transaction that reached the DDR SDRAM or local bus interfaces. The device can be configured to drive the MSRCID[0:4] MDVAL, LSRCID[0:4] and LDVAL signals, respectively on other device pins. The coding of the source ID debug information is the same as the coding of the MSTR_ID field in the AEATR register of the arbiter (See Section 6.2.6, “Arbiter Event Attributes Register (AEATR)”).

5.3.2.6.1 DDR Debug Configuration

The DDR debug configuration enables a DDR memory controller to enter debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto three optional set of pins:

- PCI pins. PCI is disabled in this mode. The external PCI host/agent must be electrically disconnected for debugging mode. Set SICRL[3] to select this mode.
- UART pins. UART operation is disabled, and any signals driven by UART devices must be electrically disconnected from the UART I/O pins. Set SICRL[5–6] to 0b01 to select this mode.

5.3.2.6.2 Local Bus Debug Configuration

The local bus debug configuration enables a LBC debug mode in which the SDRAM source ID field and data valid strobe for LBC memory accesses are driven onto UART pins. UART operation must be disabled, and any signals driven by UART devices must be electrically disconnected from the UARTI/O pins in this case. Set SICRL[5–6] to 0b11 to select this mode.

5.4 Software Watchdog Timer (WDT)

The following sections describe the theory of operation of the software watchdog timer (WDT) in the device, including a definition of the external signals and the functions they serve. Additionally, the configuration, control, and status registers are also described. Note that individual chapters in this book describe specific initialization aspects for each individual block.

5.4.1 WDT Overview

The device provides a software watchdog timer (WDT) feature to prevent system lock in case the software becomes trapped in loops with no controlled exit. Watchdog timer operations are configured in the system watchdog control register (SWCRR).

The watchdog counter is a free-running down-counter that generates a reset or a non-maskable interrupt on underflow. To prevent a reset, software must periodically restart the countdown. The WDT is responsible for asserting a hardware reset or machine-check interrupt (*mcp*) if the software fails to service the software watchdog timer for a certain period of time (for example, because software is lost or trapped in a loop with no controlled exit).

Figure 5-15 shows a high-level block diagram of the WDT.

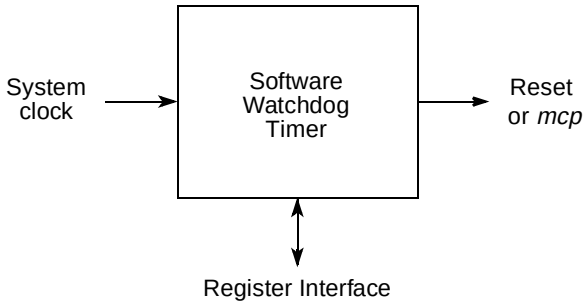


Figure 5-15. Software Watchdog Timer High-Level Block Diagram

The software watchdog timer is enabled after reset to cause a hardware reset if it times out. The user has the option of disabling the software watchdog if it is not needed. If used, the software watchdog timer requires a special service sequence to be executed periodically. Without this periodic servicing, the software watchdog timer times out and issues a reset or a non-maskable interrupt.

5.4.2 WDT Features

The WDT includes the following key features:

- Based on 16-bit prescaler and 16-bit down-counter
- Provides a selectable range for the time-out period
- Provides ~12.8-sec maximum software time-out delay for 333-MHz input clock
- Functional and programming compatibility with MPC8260 watchdog timer

5.4.3 WDT Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

  If the software watchdog timer is not needed, the user can disable it with software after a system reset. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state.
- WDT output reset/interrupt mode:

  Without software periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt (*mcp*)
- WDT prescaled/non-prescaled clock mode:

  The WDT counter clock can be prescaled by programming the SWCRR[SWPR] bit, which controls the divide-by-65,536 of the WDT counter.

5.4.4 WDT Memory Map/Register Definition

The WDT programmable register map occupies 16 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All WDT registers are 16- or 32-bits wide, located on 16-bit address boundaries, and should be accessed as 16- or 32-bit quantities. All addresses used in this chapter are offsets from the WDT base, as defined in Chapter 2, “Memory Map.”

Table 5-28 shows the WDT memory map.

Table 5-28. WDT Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x0–0x3	Reserved	—	—	—
0x4	System watchdog control register (SWCRR)	R/W	0xFFFF_0003 or 0xFFFF_0007[1]	5.4.4.1/5-25

Table 5-28. WDT Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x8	System watchdog count register (SWCNR)	R	0x0000_FFFF	5.4.4.2/5-26
0xC–0xD	Reserved	—	—	—
0xE	System watchdog service register (SWSRR)	R/W	0x0000	5.4.4.3/5-26

1 SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).

5.4.4.1 System Watchdog Control Register (SWCRR)

The system watchdog control register (SWCRR), shown in Figure 5-16, controls the software watchdog period and configures watchdog timer operation. SWCRR can be read at any time but can be written only once after system reset.

Offset 0x4 Access: Read/Write

	0				15	16						28	29	30	31
R W	SWTC					—							SWEN	SWRI	SWPR
Reset	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0[1]	1	1

1 SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).

Figure 5-16. System Watchdog Control Register (SWCRR)

Table 5-29 defines the bit fields of SWCRR.

Table 5-29. SWCRR Bit Settings

Bits	Name	Description
0–15	SWTC	Software watchdog time count The SWTC field contains the modulus that is reloaded into the watchdog counter by a service sequence. When a new value is loaded into SWCRR[SWTC], the software watchdog timer is not updated until the servicing sequence is written to the SWSRR. If SWCRR[SWEN] is loaded with 0, the modulus counter does not count. The new value is also used at the next and all subsequent reloads. Reading the SWCRR register returns the value in the system watchdog control register. Reset initializes the SWCRR[SWTC] field to 0xFFFF. **Note:** The prescaler counter is reset any time a new value is loaded into the watchdog counter and also during reset.
16–28	—	Write reserved, read = 0
29	SWEN	Watchdog enable bit Enables the watchdog timer. The reset value directly depends on the value of the RCWHR[SWEN] bit. It should be cleared by software after a system reset to disable the software watchdog timer. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state. 0 Watchdog timer disabled 1 Watchdog timer enabled **Note:** After software writes the SWRI bit, the state of SWEN cannot be changed.

Table 5-29. SWCRR Bit Settings (continued)

Bits	Name	Description
30	SWRI	Software watchdog reset/interrupt select bit A WDT timer out causes either a hard reset or machine check interrupt to the core. 0 Software watchdog timer causes a machine check interrupt to the core 1 Software watchdog timer causes a hard reset
31	SWPR	Software watchdog counter prescale bit Controls the divide-by-65,536 WDT counter prescaler 0 The WDT counter is not prescaled. 1 The WDT counter clock is prescaled.

5.4.4.2 System Watchdog Count Register (SWCNR)

The system watchdog count register (SWCNR), shown in Figure 5-17, provides visibility to the watchdog counter value. SWCNR is a read-only register. Writes to SWCNR have no effect and terminate without transfer error exception.

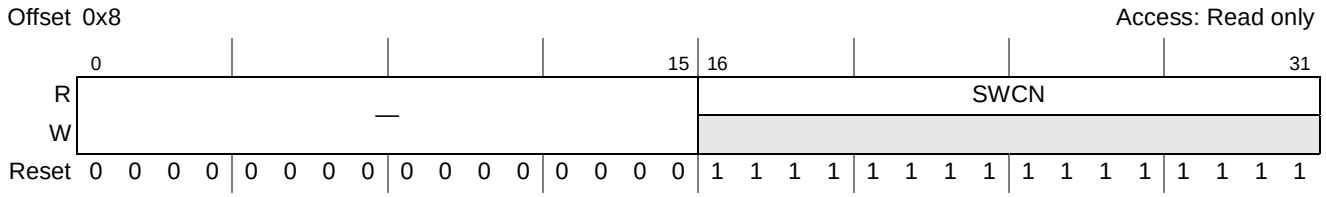


Figure 5-17. System Watchdog Count Register (SWCNR)

Table 5-30 defines the bit fields of SWCNR.

Table 5-30. SWCNR Bit Settings

Bits	Name	Description
0–15	—	Write reserved, read = 0
16–31	SWCN	Software watchdog count field. The read-only SWCNR[SWCN] field reflects the current value in the watchdog counter. Writing to the SWCNR register has no effect, and write cycles are terminated normally. Reset initializes the SWCNR[SWCN] field to 0xFFFF. **Note:** Reading the 16 least-significant bits of 32-bit SWCNR register with two 8-bit reads is not guaranteed to return a coherent value.

5.4.4.3 System Watchdog Service Register (SWSRR)

The system watchdog service register (SWSRR) is shown in Figure 5-18. When the watchdog timer is enabled, a write of 0x556C followed by a write 0xAA39 to the SWSRR register before the watchdog counter times out prevents a device reset. If the SWSRR register is not serviced before the timeout, a signal from the watchdog timer to the reset or interrupt controller module asserts a system reset or interrupt (depending on the setting of SWCRR[SWRI]).

Both writes must occur before the timeout in the order listed, but any number of instructions can be executed between the two writes. However, writing any value other than 0x556C or 0xAA39 to the SWSRR register resets the servicing sequence, requiring both values to be written to keep the watchdog

timer from causing a reset. Reset initializes the SWSRR[WS] field to 0x0000. SWSRR can be written at any time, but returns all zeros when read.

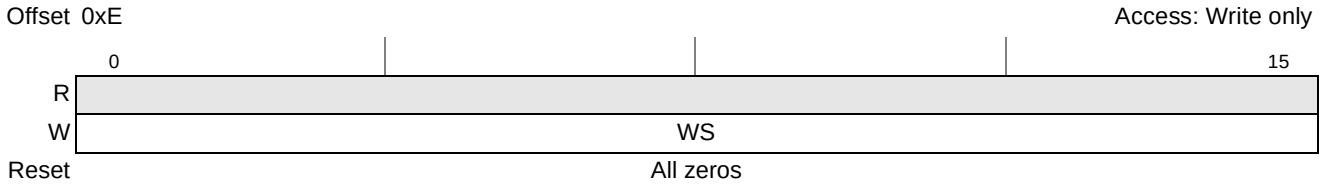


Figure 5-18. System Watchdog Service Register (SWSRR)

Table 5-31 defines the bit fields of SWCNR.

Table 5-31. SWSRR Bit Settings

Bits	Name	Description
0–15	WS	Software watchdog service field. The user should periodically write 0x556C followed by 0xAA39 to this register to prevent a software watchdog timer timeout. SWSRR[WS] can be written at any time, but returns all zeros when read.

5.4.5 Functional Description

5.4.5.1 Software Watchdog Timer Unit

The device provides a software watchdog timer (WDT) feature to prevent system lock in case the software becomes trapped in loops with no controlled exit. Watchdog timer operations are configured in the system watchdog control register (SWCRR).

The software watchdog timer is enabled after reset to cause a soft reset or non-maskable interrupt (MCP) if it times out. If the software watchdog timer is not needed, the user must clear SWCRR[SWEN] to disable it. If used, the software watchdog timer requires a special service sequence to be executed periodically. Without this periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt, as programmed in SWCRR[SWRI]. Once software writes SWRI, the state of SWEN cannot be changed.

The software watchdog timer service sequence consists of the following two steps:

- Write 0x556C to the system watchdog service register (SWSRR)
- Write 0xAA39 to SWSRR

The service sequence reloads the watchdog timer and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the SWSRR, the entire sequence must start over. Although the writes must occur in the correct order before a time-out, any number of instructions can be executed between the writes. This allows interrupts and exceptions to occur between the two writes when necessary. Figure 5-19 shows a state diagram for the watchdog timer.

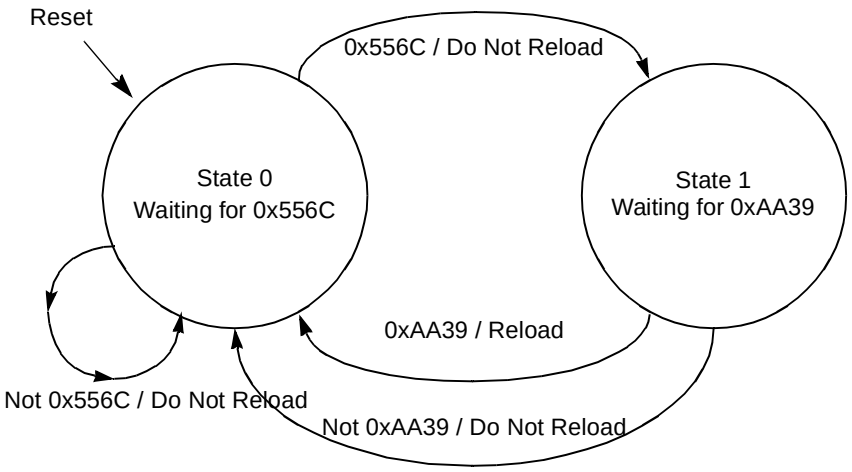


Figure 5-19. Software Watchdog Timer Service State Diagram

Although most software disciplines permit or even encourage the watchdog concept, some systems require a selection of time-out periods. For this reason, the software watchdog timer must provide a selectable range for the time-out period. Figure 5-20 shows how to handle this need.

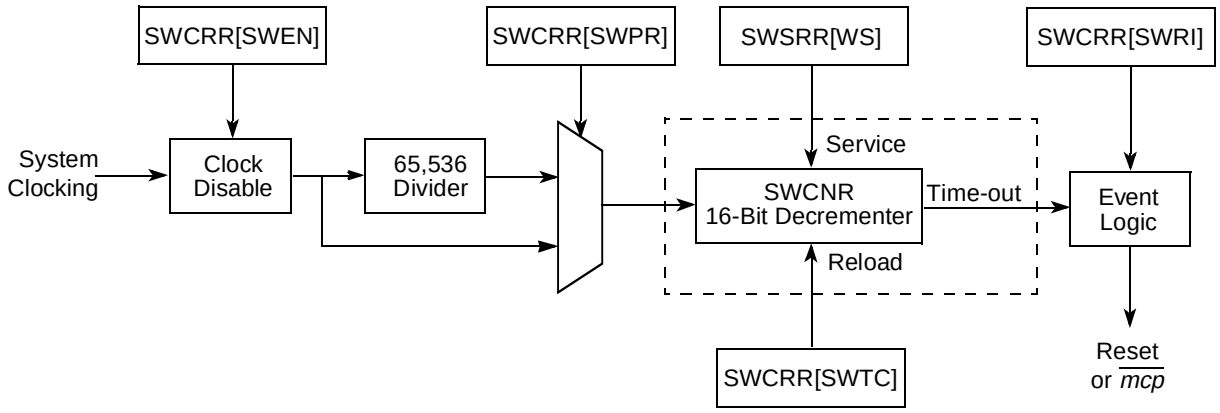


Figure 5-20. Software Watchdog Timer Functional Block Diagram

Figure 5-20 shows that the range is determined by SWCRR[SWTC]. The value in SWTC is then loaded into a 16-bit decrementer clocked by the system clock. An additional divide-by-65,536 prescaler value is used when needed.

The decrementer begins counting when loaded with a value from SWTC. After the timer reaches 0x0, a software watchdog expiration request is issued to the reset or *mcp* (machine check) control logic. Upon reset, SWTC is set to the maximum value and is again loaded into the system watchdog service register (SWSRR), starting the process over. When a new value is loaded into SWTC, the software watchdog timer is not updated until the servicing sequence is written to the SWSRR. If SWCRR[SWEN] is loaded with 0, the modulus counter does not count.

5.4.5.2 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

  If the software watchdog timer is not needed, the user can disable it. The SWCRR[SWEN] bit enables the watchdog timer. It should be cleared by software after a system reset to disable the software watchdog timer. When the watchdog timer is disabled, the watchdog counter and prescaler counter are held in a stopped state.

  — WDT enable mode (SWCRR[SWEN] = 1)

  This is the default value after soft reset.

  — WDT disable mode (SWCRR[SWEN] = 0)

- WDT reset/interrupt output mode

  Without software periodic servicing, the software watchdog timer times out and issues a reset or a nonmaskable interrupt (*mcp*), programmed in SWCRR[SWRI].

  According to the value of SWCRR[SWRI], the WDT timer causes a hard reset or machine check interrupt to the core.

  — Reset mode (SWCRR[SWRI] = 1).

  Software watchdog timer causes a hard reset (this is the default value after hard reset).

  — Interrupt mode (SWCRR[SWRI] = 0).

  Software watchdog timer causes a machine check interrupt to the core.

- WDT prescaled/non-prescaled clock mode

  The WDT counter clock can be prescaled by programming the SWCRR[SWPR] bit that controls the divide-by-65,536 of the WDT counter.

  — Prescale mode (SWCRR[SWPR] = 1)

  The WDT clock is prescaled.

  — Non-prescale mode (SWCRR[SWPR] = 0)

  The WDT clock is not prescaled.

5.4.6 Initialization/Application Information

5.4.6.1 WDT Programming Guidelines

The software watchdog timer is enabled (by the default value of SWCRR[SWEN]) after reset. The following initialization sequence of WDT is required:

- WDT disabling

  If the software watchdog timer is not needed, the user must clear SWCRR[SWEN] bit to disable the WDT not later than its timer times out (~12.8 sec. for a 333-MHz system clock).

- WDT initial servicing

  If the software watchdog timer is to be used, the special service sequence, described in Section 5.4.5.1, “Software Watchdog Timer Unit,” must be executed after system reset and not later than the first WDT time-out (~12.8 sec. for a 333-MHz system clock).

  Subsequently, periodical WDT servicing should be performed according to the programming guidelines given in Section 5.4.5.1, “Software Watchdog Timer Unit.”

5.5 Real Time Clock Module (RTC)

The following sections describe the theory of operation of the real time clock module (RTC) including a definition of the external signals and the functions it serves. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this reference manual describe additional specific initialization aspects for each individual block.

5.5.1 RTC Overview

The device platform provides a real time clock (RTC) timer suitable for timestamping or time and calendar generation. It can maintain a one-second count which is unique over a period of approximately 136 years.

The RTC can be initialized by software with an initial count value using the real time counter load register (RTLDR). It can also be programmed to generate an interrupt every second. The real time counter control register (RTCTR) is used to enable or disable the various timer functions. The real time counter event register (RTEVR) is used to report the interrupt source. The RTC counter is initialized by software and can be disabled if needed.

Figure 5-21 shows the high level RTC block diagram.

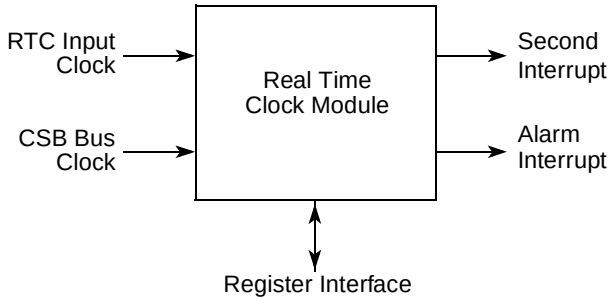


Figure 5-21. Real Time Clock Module High Level Block Diagram

5.5.2 RTC Features

The key features of the RTC include the following:

- Maintains a one-second count, unique over a period of thousand of years
- 32-bit RTC counter can be initialized by software to specific initial count value
- Provides an alarm function with programmable and maskable alarm interrupt
- Provides programmable and maskable every second interrupt

- Uses two possible clock sources: the CSB bus clock or an external RTC clock
- RTC function can be disabled if needed

5.5.3 RTC Modes of Operation

The RTC unit can operate in the following modes:

- RTC enable/disable mode
- RTC every-second interrupt enable/disable mode
- RTC alarm interrupt enable/disable mode
- RTC internal/external input clock mode

5.5.4 RTC External Signal Description

This section provides an overview and detailed descriptions of the RTC signals.

There is one distinct external RTC clock input signal, defined in Table 5-32.

Table 5-32. RTC Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
RTC_CLK	RTC_CLK	Real time clock input.	I	N/A	—

Table 5-33 provides a detailed description of the external RTC signal.

Table 5-33. RTC External Signal—Detailed Signal Description

Signal	I/O	Description	
RTC_CLK	I	This signal is used as the timebase for the real time clock module.	
		State Meaning	—
		Timing	—

5.5.5 RTC Memory Map/Register Definition

The RTC programmable register map occupies 32 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All RTC registers are 32 bits wide that are located on 32-bit address boundaries and should only be accessed as a 32-bit quantities.

All addresses used in this section are offsets from the RTC base, as defined in Chapter 2, “Memory Map.”

Table 5-28 shows the memory map of the RTC.

Table 5-34. RTC Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Real time counter control register (RTCNR)	R/W	0x0000_0000	5.5.5.1/5-32
0x04	Real time counter load register (RTLDR)	R/W	0x0000_0000	5.5.5.2/5-33
0x08	Real time counter prescale register (RTPSR)	R/W	0x0000_0000	5.5.5.3/5-33
0x0C	Real time counter register (RTCTR)	R	0x0000_0000	5.5.5.4/5-34
0x10	Real time counter event register (RTEVR)	w1c	0x0000_0000	5.5.5.5/5-34
0x14	Real time counter alarm register (RTALR)	R/W	0xFFFF_FFFF	5.5.5.6/5-35
0x18–0x1F	Reserved	—	—	—

5.5.5.1 Real Time Counter Control Register (RTCNR)

The real time counter control register (RTCNR), shown in Figure 5-22, is used to enable RTC functions. The register can be read at any time.

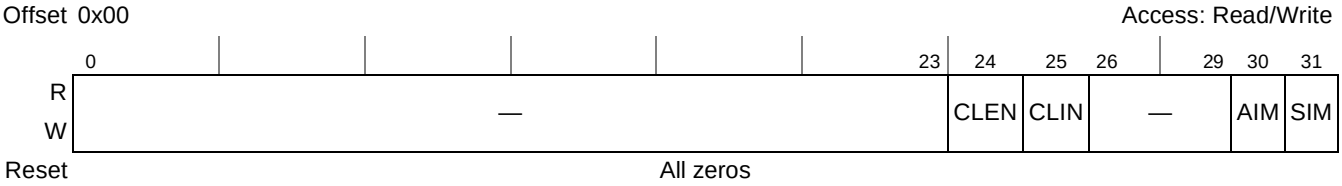


Figure 5-22. Real Time Counter Control Register (RTCNR)

Table 5-35 defines the bit fields of RTCNR.

Table 5-35. RTCNR Bit Settings

Bits	Name	Description
0–23	—	Write reserved, read = 0
24	CLEN	Clock enable control bit. This bit controls the counting of the RTC. When the RTC's clock is disabled, the counter maintains its old value. When the counter's clock is enabled, it continues counting using the previous value. 0 Disable counter. 1 Enable counter.
25	CLIN	Input clock control bit. The input clock to the RTC may be either the CSB clock or an external RTC clock. 0 The input clock to the periodic interrupt timer is CSB input clock. 1 The input clock to the periodic interrupt timer is the external RTC clock.
26–29	—	Write reserved, read = 0

Table 5-35. RTCNR Bit Settings (continued)

Bits	Name	Description
30	AIM	Alarm interrupt mask bit. Used to enable or disable (mask) the RTC alarm interrupt when the RTC's 32-bit counter reaches RTALR[ALR] value. 0 Alarm interrupt generation disabled. 1 Alarm interrupt generation enabled.
31	SIM	Second interrupt mask bit. Used to enable or disable (mask) the RTC periodic interrupt. 0 Periodic interrupt generation disabled. 1 Periodic interrupt generation enabled.

5.5.5.2 Real Time Counter Load Register (RTLDR)

The real time counter load register (RTLDR), shown in Figure 5-23, contains the 32-bit value to be loaded in the 32-bit RTC counter.

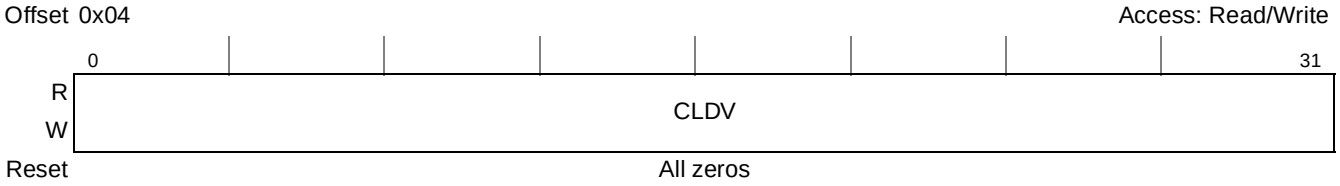


Figure 5-23. Real Time Counter Load Register (RTLDR)

Table 5-36 defines the bit fields of RTLDR.

Table 5-36. RTLDR Bit Settings

Bits	Name	Description
0–31	CLDV	Contains the 32-bit value to be loaded in the 32-bit RTC counter.

5.5.5.3 Real Time Counter Prescale Register (RTPSR)

The real time counter prescale register (RTPSR), shown in Figure 5-24, is a read/write register used to configure the RTC prescaler's value.

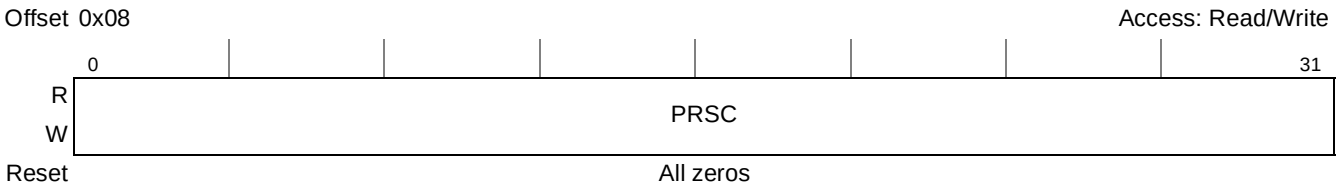


Figure 5-24. Real Time Counter Prescale Register (RTPSR)

Table 5-37 defines the bit fields of RTPSR.

Table 5-37. RTPSR Bit Settings

Bits	Name	Description
0–31	PRSC	RTC prescaler bits. Select the input clock divider for the RTC counter clock. The prescaler is programmed to divide the RTC clock input by values from 1 to 4,294,967,296. The value 0x0000 divides the clock by 1 and 0xFFFF_FFFF divides the clock by 4,294,967,296. To accurately predict the timing of the next count, change the RTPSR[PRSC] field only when the enable bit RTCNR[CLE] is clear. Changing the RTPSR[PRSC] bits resets the prescaler counter. System reset and the loading of a new value into the counter both reset the prescaler counter. Clearing RTCNR[CLE] stops the prescaler counter.

5.5.5.4 Real Time Counter Register (RTCTR)

The real time counter register (RTCTR), shown in Figure 5-25, is a read-only register that shows the current value in the RTC counter.

The CNTV value is not affected by reads or writes to RTCTR.

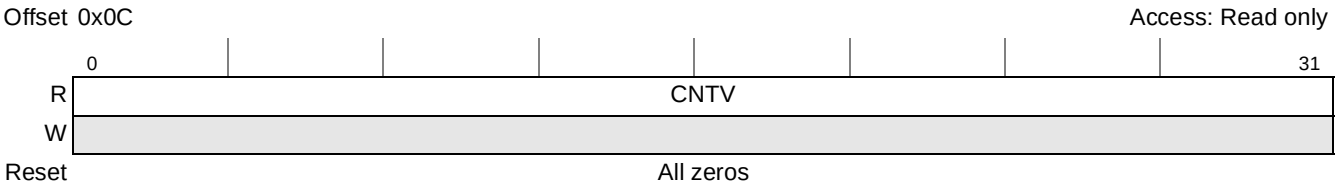


Figure 5-25. Real Time Counter Register (RTCTR)

Table 5-38 defines the bit fields of RTCTR.

Table 5-38. RTCTR Bit Settings

Bits	Name	Description
0–31	CNTV	RTC counter value field. RTCTR[CNTV] contains the current value of the time counter. This is a read-only field. Writes have no effect on RTCTR[CNTV].

5.5.5.5 Real Time Counter Event Register (RTEVR)

The real time counter event register (RTEVR), shown in Figure 5-26, is used to report the source of the interrupts. The register can be read at any time.

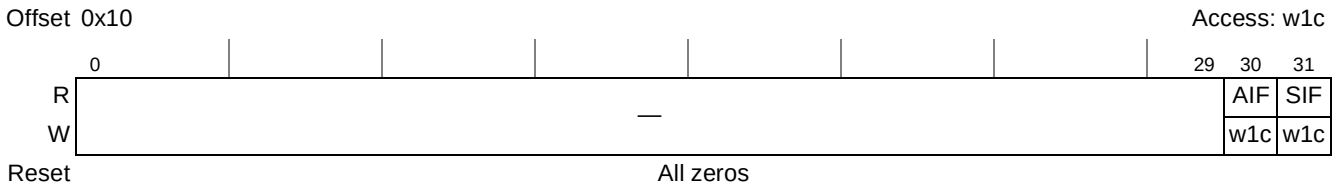


Figure 5-26. Real Time Counter Event Register (RTEVR)

RTEVR bits are cleared by writing ones. Writing zeros does not affect the value of the status bits.

Table 5-39 defines the bit fields of RTEVR.

Table 5-39. RTEVR Bit Settings

Bits	Name	Description
0–29	—	Write reserved, read = 0
30	AIF	Alarm interrupt flag bit. Used to indicate the alarm interrupt. It is set if the RTC issues an interrupt after the RTC counter counts to zero.
31	SIF	Second interrupt flag bit. Used to indicate the every-second interrupt. This status bit is set each time that the prescaler count reaches zero and should be cleared by software.

5.5.5.6 Real Time Counter Alarm Register (RTALR)

The real time counter alarm register (RTALR), shown in Figure 5-27, contains the 32-bit alarm (ALRM) value. When the value of the RTC counter equals the RTALR[ALRM] value, a maskable interrupt is generated.

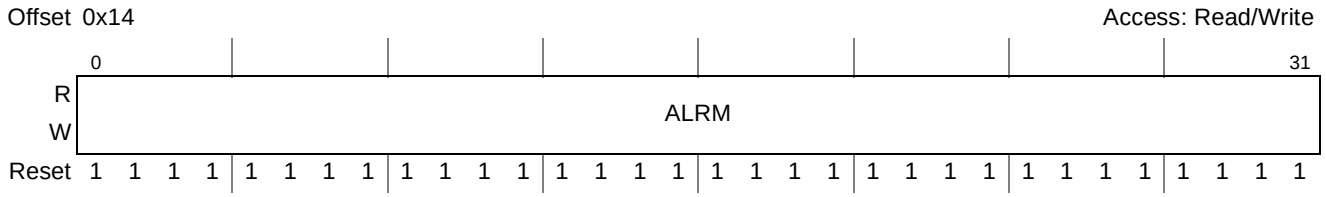


Figure 5-27. Real Time Counter Alarm Register (RTALR)

Table 5-40 defines the bit fields of RTALR.

Table 5-40. RTALR Bit Settings

Bits	Name	Description
0–31	ALRM	RTC alarm value. The alarm interrupt is generated when the value of the RTC counter equals RTALR[ALRM].

5.5.6 Functional Description

5.5.6.1 Real Time Counter Unit

The real time clock (RTC) timer is suitable for time stamping or time and calendar generation. It can maintain a one-second count which is unique over a period of approximately 136 years. Software can convert this count into time-of-day or calendar information if required. An alarm function is also provided. The RTC can be clocked by the internal system bus clock or by an external clock source. The RTC consists of 32-bit up-counter which is incremented by an one-second count clock derived from the RTC input clock. The RTC can be programmed to generate a maskable interrupt when the time value matches the value in its associated alarm register.

The RTC can be initialized by software with an initial count value in the real time counter load register (RTLDR). It can also be programmed to generate an interrupt every second. The real time counter control

register (RTCTR) is used to enable or disable the various timer functions. The real time counter event register (RTEVR) is used to report the interrupt source. The RTC counter is reset to zero on hard reset but is not affected by soft reset. It is initialized by the software. The RTC function can be disabled.

Figure 5-28 shows the functional RTC block diagram.

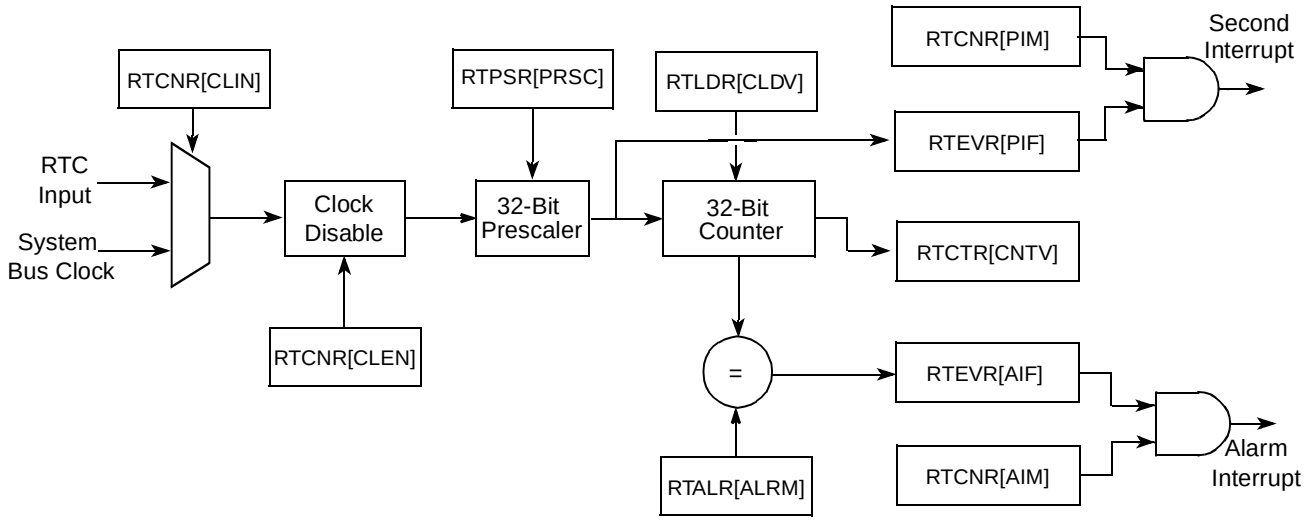


Figure 5-28. Real Time Clock Module Functional Block Diagram

5.5.6.2 RTC Operational Modes

The RTC unit can operate in the following modes:

- RTC enable/disable mode:

  RTCNR[CLEN] enables the RTC timer. It should be set by software after a system reset to enable the RTC timer.

  — RTC disable mode (RTCNR[CLEN] = 0)

  When the RTC's clock is disabled, counter maintains its old value (default).

  — RTC enable mode (RTCNR[CLEN] = 1)

  When the counter's clock is enabled, it continues counting using the previous value.

- RTC every-second interrupt enable/disable mode:

  — RTC every-second interrupt enable mode (RTCNR[SIM] = 1)

  In this mode the RTC set the RTEVR[SIF] flag and generate an interrupt after the RTC's 32-bit counter reaches zero.

  — RTC every-second interrupt disable mode (RTCNR[SIM] = 0)

  In this mode the RTC sets the RTEVR[SIF] flag but does not generate an interrupt after the RTC's 32-bit counter reaches zero.

- RTC alarm interrupt enable/disable mode:

  — RTC alarm interrupt enable mode (RTCNR[AIM] = 1)

  In this mode, the RTC sets the RTEVR[AIF] flag and generates an interrupt each time when the RTC's 32-bit counter reaches the RTALR[ALR] value.

— RTC alarm interrupt disable mode (RTCNR[AIM] = 0)

In this mode the RTC sets the RTEVR[AIF] flag but does not generate an interrupt when the RTC's 32-bit counter reaches the RTALR[ALR] value.

- RTC internal/external input clock mode:

The input clock to the RTC may be the CSB clock or an external 32.768-kHz crystal.

— RTC uses the internal input clock mode (RTCNR[CLIN] = 0)

— RTC uses the external 32.768-kHz crystal clock (RTCNR[CLIN] = 1)

5.5.7 RTC Programming Guidelines

The following initialization sequence for the RTC is recommended:

1. Write to RTPSR to set the RTC prescaler to the desired value
2. Write to RTLDR to initialize the RTC initial value
3. Write to RTALR to program the RTC alarm value, if needed
4. Write to RTCNR to configure and start the RTC operation: RTC input clock source, second/alarm interrupt mask, RTC clock enable.

5.6 Periodic Interval Timer (PIT)

The following sections describe theory of operation of the periodic interval timer (PIT) including a definition of the external signals and the functions it serves. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this reference manual describe additional specific initialization aspects for each individual block.

5.6.1 PIT Overview

The periodic interval timer (PIT) that generates periodic interrupts for a real-time operating system or an application software.

The PIT consists of a 32-bit down-counter which is decremented by a clock derived from a CSB clock or from an external 32.768-kHz crystal. The 32-bit counter decrements to zero when loaded with a initial value from the periodic interval timer load register (PTLDR). The periodic interval timer control register (PTCTR) is used to enable or disable the various timer functions. The periodic interval timer event register (PTEVR) is used to report the interrupt source. The PIT function can be disabled if needed.

Figure 5-29 shows the functional PIT block diagram.

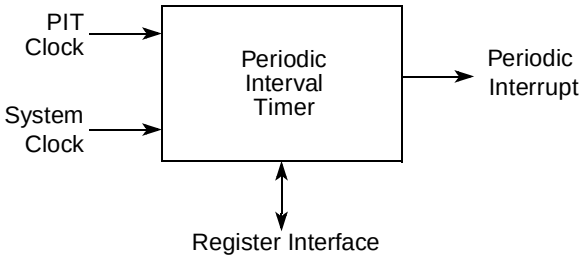


Figure 5-29. Periodic Interval Timer High Level Block Diagram

5.6.2 PIT Features

The key features of the PIT include the following:

- Maintains a 32-bit down-counter, clocked by a 32-bit prescaled input clock
- 32-bit PIT counter can be initialized by software to specific initial count value
- Provides programmable and maskable periodic interrupt
- Uses two possible clock sources: the CSB clock or an external PIT clock
- PIT function can be disabled

5.6.3 PIT Modes of Operation

The PIT unit can operate in the following modes:

- PIT enable/disable mode
- PIT periodic interrupt enable/disable mode
- PIT internal/external input clock mode

5.6.4 PIT External Signal Description

This section provides an overview and detailed descriptions of the PIT signals.

There is one distinct external input signal (PIT clock), defined in Table 5-41.

Table 5-41. PIT Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
PIT_CLK	PIT_CLK	Periodic interval timer.	I	N/A	—

Table 5-42 describes of the external PIT signal.

Table 5-42. PIT External Signal—Detailed Signal Descriptions

Signal	I/O	Description	
PIT_CLK	I	This signal is used as the timebase for the periodic interval timer module.	
		State Meaning	—
		Timing	—

5.6.5 PIT Memory Map/Register Definition

The PIT programmable register map occupies 32 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All PIT registers are 32 bits wide and reside on 32-bit address boundaries and should only be accessed as 32-bit quantities.

All addresses used in this chapter are offsets from PIT base, as defined in Chapter 2, “Memory Map.”

Table 5-43 shows the PIT memory map.

Table 5-43. PIT Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Periodic interval timer control register (PTCNR)	R/W	0x0000_0000	5.6.5.1/5-39
0x04	Periodic interval timer load register (PTLDR)	R/W	0x0000_0000	5.6.5.2/5-40
0x08	Periodic interval timer prescale register (PTPSR)	R/W	0x0000_0000	5.6.5.3/5-41
0x0C	Periodic interval timer counter register (PTCTR)	R	0x0000_0000	5.6.5.4/5-41
0x10	Periodic interval timer event register (PTEVR)	w1c	0x0000_0000	5.6.5.5/5-41
0x14–0x1F	Reserved	—	—	—

5.6.5.1 Periodic Interval Timer Control Register (PTCNR)

The periodic interval timer control register (PTCNR), shown in Figure 5-30, is used to enable the different PIT functions. The register can be read at any time.

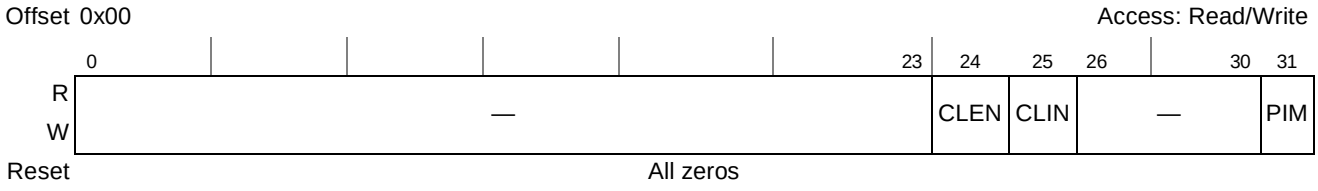


Figure 5-30. Periodic Interval Timer Control Register (PTCNR)

Table 5-44 defines the bit fields of PTCNR.

Table 5-44. PTCNR Bit Settings

Bits	Name	Description
0–23	—	Write reserved, read = 0
24	CLEN	Clock enable control bit. Controls the counting of the PIT. When the PIT's clock is disabled, the counter maintains its old value. When the counter's clock is enabled, it continues counting using the previous value. 0 Disable counter. 1 Enable counter.
25	CLIN	Input clock control bit. The input clock to the PIT can be either an internal system clock or an external PIT clock. 0 The input clock to the periodic interrupt timer is internal system clock. 1 The input clock to the periodic interrupt timer is external PIT clock.
26–30	—	Write reserved, read = 0
31	PIM	Periodic interrupt mask bit. Used to enable or disable (mask) the PIT periodic interrupt. 0 Periodic interrupt generation disabled. 1 Periodic interrupt generation enabled.

5.6.5.2 Periodic Interval Timer Load Register (PTLDR)

The periodic interval timer load register (PTLDR), shown in Figure 5-31, contains the 32-bit value to be loaded in a 32-bit PIT counter.

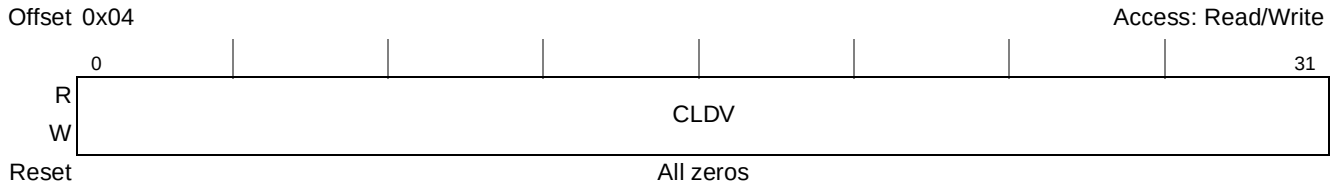


Figure 5-31. Periodic Interval Timer Load Register (PTLDR)

Table 5-45 defines the bit fields of PTLDR.

Table 5-45. PTLDR Bit Settings

Bits	Name	Description
0–31	CLDV	Contains the 32-bit value to be loaded in a 32-bit PIT counter.

5.6.5.3 Periodic Interval Timer Prescale Register (PTPSR)

The periodic interval timer prescale register (PTPSR), shown in Figure 5-32, is a read/write register that used to configure the PIT prescaler's value.

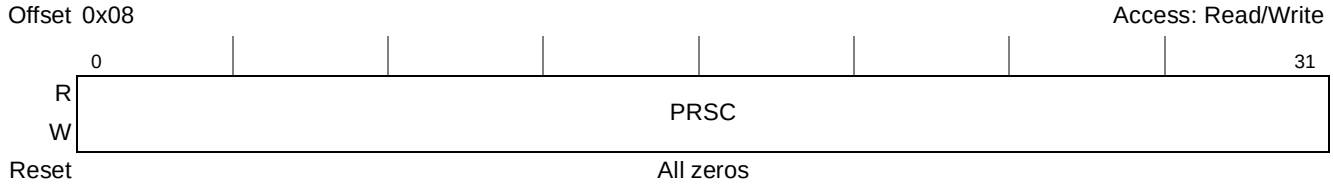


Figure 5-32. Periodic Interval Timer Prescale Register (PTPSR)

Table 5-46 defines the bit fields of PTPSR.

Table 5-46. PTPSR Bit Settings

Bits	Name	Description
0–31	PRSC	PIT prescaler bits. Selects the input clock divider to generate the PIT counter clock. The prescaler is programmed to divide the PIT clock input by values from 1 to 4,294,967,296. The value 0x0000 divides the clock by 1 and 0xFFFF_FFFF divides the clock by 4,294,967,296. To accurately predict the timing of the next count, change the PRSC bit only when the enable bit PTCNR[CLE] is clear. Changing PRSC resets the prescaler counter. System reset and the loading of a new value into the counter also reset the prescaler counter. Clearing the PTCNR[CLE] bit stops the prescaler counter.

5.6.5.4 Periodic Interval Timer Counter Register (PTCTR)

The periodic interval timer counter register (PTCTR), shown in Figure 5-33, is a read-only register that shows the current value in the PIT counter. The PTCTR counter is not affected by reads or writes.

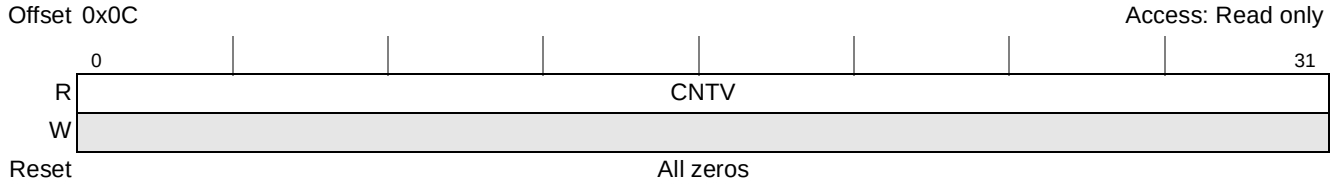


Figure 5-33. Periodic Interval Timer Counter Register (PTCTR)

Table 5-47 defines the bit fields of PTCTR.

Table 5-47. PTCTR Bit Settings

Bits	Name	Description
0–31	CNTV	PIT counter value field. Contains the current value of the time counter. This is a read-only field. Writes have no effect on PTCTR[CNTV].

5.6.5.5 Periodic Interval Timer Event Register (PTEVR)

The periodic interval timer event register (PTEVR), shown in Figure 5-34, is used to report the source of the interrupts. The register can be read at any time.

PTEVR bits are cleared by writing ones. Writing zeros does not affect the value of the status bits.

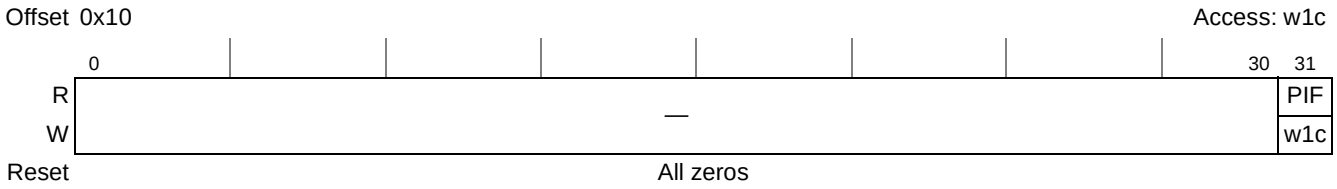


Figure 5-34. Periodic Interval Timer Event Register (PTEVR)

Table 5-48 defines the bit fields of PTEVR.

Table 5-48. PTEVR Bit Settings

Bits	Name	Description
0–30	—	Write reserved, read = 0
31	PIF	Periodic interrupt flag bit. Used to indicate the periodic interrupt. Its asserted if the PIT issues an interrupt after the SPMPIT counter counts to zero. This status bit should be cleared by software.

5.6.6 Functional Description

5.6.6.1 Periodic Interval Timer Unit

The PIT generates periodic interrupts for use with a real-time operating system or the application software. It consists of a 32-bit down-counter which is decremented by a clock derived from the CSB clock or from the PIT clock. The 32-bit counter decrements to zero when loaded with a initial value from the periodic interval timer load register (PTLDR). After the timer reaches zero, PTEVR[PIF] is set and an interrupt is generated if PTCNR[PIM] = 1. At the next count cycle, the value in the PTLDR[CLDV] is loaded into the counter and the process repeats. When a new value is loaded into the PTLDR[CLDV], the PIT is updated, the prescaler counter is reset, and the counter begins counting. Setting of PTEVR[PIF] generates an interrupt, that remains pending until PTEVR[PIF] is cleared. If PTEVR[PIF] is set again before being cleared, the interrupt remains pending until PTEVR[PIF] is cleared. Any write to the PTLDR[CLDV] stops the current countdown and the count resumes with the new value in PTLDR[CLDV]. If PTCNR[CLEN] = 0, the PIT cannot count and retains the old count value. PTCTR contain the PIT current value. The PIT function can be disabled if needed.

Figure 5-35 shows the functional PIT block diagram.

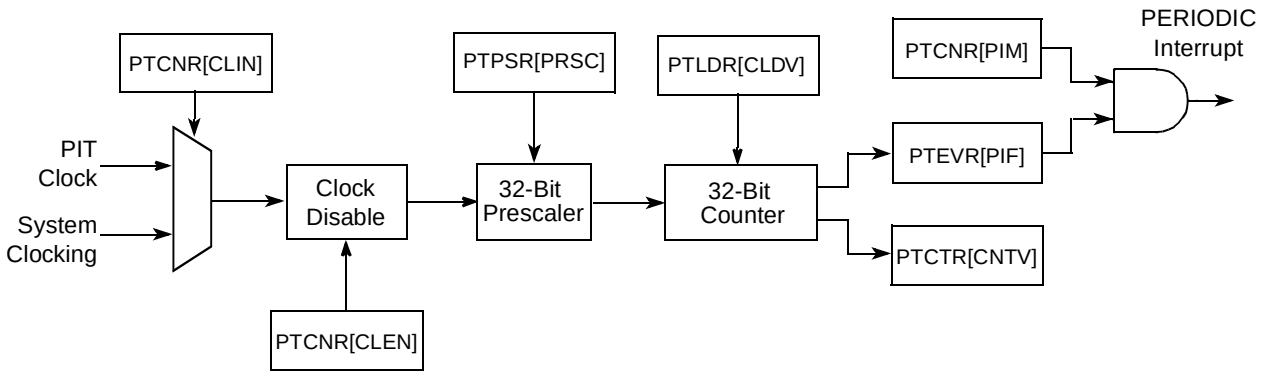


Figure 5-35. Periodic Interval Timer Functional Block Diagram

5.6.6.2 PIT Operational Modes

The PIT unit can operate in the following modes:

- PIT enable/disable mode:

  The PTCNR[CLEN] bit enables the PIT timer. It should be set by software after a system reset to enable the PIT timer.

  — PIT disable mode (PTCNR[CLEN] = 0). When the PIT's clock is disabled, counter maintains its old value.

  — PIT enable mode (PTCNR[CLEN] = 1). When the counter's clock is enabled, it continues counting using the previous value.
- PIT periodic interrupt enable/disable mode:

  — PIT periodic interrupt enable mode (PTCNR[PIM] = 1). After the PIT's 32-bit counter reaches zero, the PIT sets the PTEVR[PIF] flag and generates an interrupt.

  — PIT periodic interrupt disable mode (PTCNR[PIM] = 0). After the PIT's 32-bit counter reaches zero, the PIT sets the PTEVR[PIF] flag but does not generate an interrupt.
- PIT internal/external input clock mode:

  The input clock to the PIT may be an internal system clock or the PIT clock.

  — PIT use the internal input clock mode (PTCNR[CLIN] = 0)

  — PIT use the PIT clock (PTCNR[CLIN] = 1)

5.6.7 PIT Programming Guidelines

The following initialization sequence of PIT is recommended:

1. Write to PTPSR to set the PIT prescaler to the desired value
2. Write to PTLDR to initialize the PIT initial value
3. Write to PTCNR to configure and start the PIT operation: PIT input clock source, periodic interrupt mask, PIT clock enable.

See Section 5.5.7, "RTC Programming Guidelines," for real-time clock programming guidelines.

5.7 General-Purpose Timers (GTMs)

The following sections describe theory of operation of the general purpose (global) timer module, including a definition of the external signals and the functionality. Additionally, the configuration, control, and status registers are described. Note that individual chapters in this book describe additional specific initialization aspects for each individual block.

5.7.1 GTM Overview

Each global timer module (GTM) includes four identical 16-bit general-purpose timers, two 32-bit timers or one 64-bit timer. Each GTM timer consists of a timer prescale register (GTPSR), a timer mode register (GTMDR) a timer capture register (GTCPR), a timer counter register (GTCNR), a timer reference register

(GTRFR), a timer event register (GTEVR), and a timer global configuration register (GTCFR). The GTPSRs and the GTMDRs contain the primary and secondary prescalers, programmed by the user.

Figure 5-36 shows the functional GTM block diagram.

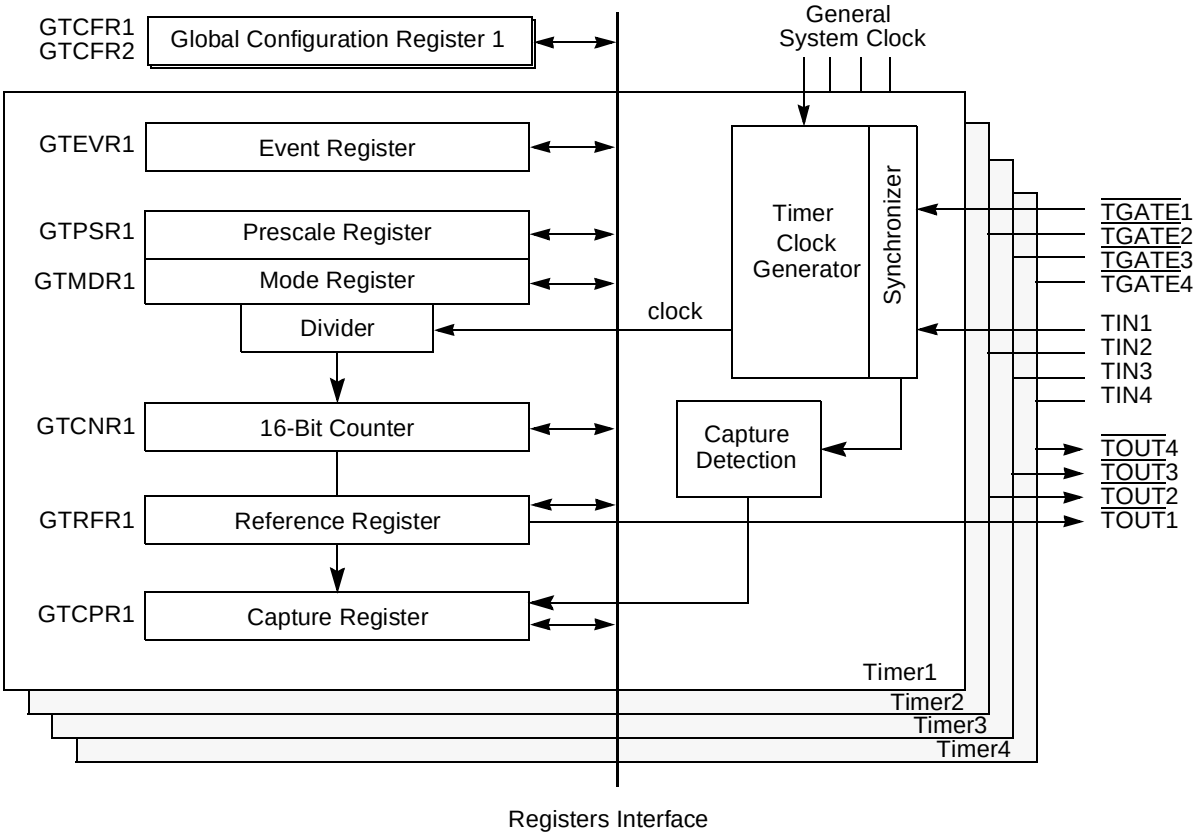


Figure 5-36. Global Timers Block Diagram

5.7.2 GTM Features

The key features of the timer include the following:

- The maximum input clock is the system bus clock
- Four 16-bit programmable timers
- Two timers cascaded internally or externally to form a 32-bit timer
- One timer cascaded internally or externally to form a 64-bit timer
- Maximum period of ~50 msecond (at 333-MHz bus clock and prescaler=256) for 16-bit timer
- Maximum period of ~12.8 seconds (at 333-MHz bus clock and prescaler=256) for 32-bit timer
- Maximum period of thousands of years (at 333-MHz bus clock and prescaler=256) for 64-bit timer
- 3-nanosecond timer resolution (at 333-MHz bus clock and no prescaler)
- Resolution and maximum period can be traded off by selecting prescaler divisor
- Three programmable input clock sources for the timer prescalers
- Input capture capability

- Output compare with programmable mode for the output pin
- Free run and restart modes
- Functional and programming compatibility with MPC8260 timers

5.7.3 GTM Modes of Operation

The GTM unit can operate in the following modes.

5.7.3.1 Cascaded Modes

GTCFR*n*[PCAS] and GTCFR2[SCAS] are used to put the timers into different cascaded modes:

- Non-cascaded mode: Each timer (timer 1, timer 2, timer 3, and timer 4), function as a independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR, and GTCNR. In this mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with corresponding 16-bit bus cycles.
- Pair-cascaded mode: In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 can be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer, or two 32-bit timers. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.
- Super-cascaded mode: In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

5.7.3.2 Clock Source Modes

The clock input to the timer’s prescaler can be selected from three sources:

- The system clock
- The system slow go clock (system bus clock internally divided by 16)
- The corresponding TINx pin

5.7.3.3 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding GTMRR selects each mode.

- Free run reference mode. The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode. The corresponding timer count is reset immediately after the reference value is reached.

5.7.3.4 Capture Modes

Each timer has a 16-bit field in GTCPR, used to latch the value of the counter when a defined transition of TINx is sensed by the corresponding input capture edge detector.

- Normal gate mode enables the count on a falling edge of the $\overline{\text{TGATE}}$ pin and disables the count on the rising edge of $\overline{\text{TGATE}}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}}$.
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of the $\overline{\text{TGATE}}$ pin. This mode has applications in pulse interval measurement and bus monitoring.

5.7.4 GTM External Signal Description

This section provides an overview and detailed descriptions of the GTM signals.

There are four distinct external input timer capture signals (TIN1, TIN2, TIN3, and TIN4), four distinct external input timer get signals ($\overline{\text{TGATE1}}$, $\overline{\text{TGATE2}}$, $\overline{\text{TGATE3}}$, and $\overline{\text{TGATE4}}$), and four distinct external timer output signals ($\overline{\text{TOUT1}}$, $\overline{\text{TOUT2}}$, $\overline{\text{TOUT3}}$, and $\overline{\text{TOUT4}}$). The GTM interface signals are defined in Table 5-49.

Table 5-49. GTM Signal Properties

Name	Port	Function	I/O	Reset	Require Pull Up
TIN1	TIN1	Global timer 1 capture control signal	I	0	No
TIN2	TIN2	Global timer 2 capture control signal	I	0	No
TIN3	TIN3	Global timer 3 capture control signal	I	0	No
TIN4	TIN4	Global timer 4 capture control signal	I	0	No
$\overline{\text{TGATE1}}$	$\overline{\text{TGATE1}}$	Global timer 1 counter gate control signal	I	0	No
$\overline{\text{TGATE2}}$	$\overline{\text{TGATE2}}$	Global timer 2 counter gate control signal	I	0	No
$\overline{\text{TGATE3}}$	$\overline{\text{TGATE3}}$	Global timer 3 counter gate control signal	I	0	No
$\overline{\text{TGATE4}}$	$\overline{\text{TGATE4}}$	Global timer 4 counter gate control signal	I	0	No
$\overline{\text{TOUT1}}$	$\overline{\text{TOUT1}}$	Global timer 1 counter output signal	O	1	No
$\overline{\text{TOUT2}}$	$\overline{\text{TOUT2}}$	Global timer 2 counter output signal	O	1	No
$\overline{\text{TOUT3}}$	$\overline{\text{TOUT3}}$	Global timer 3 counter output signal	O	1	No
$\overline{\text{TOUT4}}$	$\overline{\text{TOUT4}}$	Global timer 4 counter output signal	O	1	No

Table 5-50 provides detailed descriptions of the external GTM signals.

Table 5-50. GTM External Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TIN*n*	I	Global timer capture control signal. Used to latch the value of the counter when a defined transition of TIN*n* is sensed by the corresponding input capture edge detector.	
		State Meaning	Asserted/Negated —According to the programmed polarity by the corresponding GTMDR*n*[CE]). Each timer has a 16-bit GTCPR used to latch the value of the counter when a defined transition of TIN*n* is sensed by the corresponding input capture edge detector. Upon a capture or reference event, the corresponding GTEVR bit is set and a maskable interrupt request is issued to the interrupt controller.
		Timing	Assertion/Negation—Asynchronous to internal bus clock. TIN*n* is internally synchronized to the system bus clock. If TIN*n* meets the asynchronous input setup time, the value of counter is captured after one system bus clock when working with the internal clock.
$\overline{\text{TGATE}}n$	I	Global timer counter gate control signal. Used to gate/restart the counter when a defined transition of $\overline{\text{TGATE}}n$ is sensed by the corresponding input capture edge detector.	
		State Meaning	Asserted/Negated—According to the programmed polarity by the corresponding GTCFR[GMx] bits. In a reset gate mode (GTCFR[GM*n*] = 0), the $\overline{\text{TGATE}}n$ pin is used to enable/disable count. A falling $\overline{\text{TGATE}}n$ pin enables and restarts the count and a rising edge of $\overline{\text{TGATE}}n$ disables the count. In a normal gate mode (GTCFR[GM*n*] = 1), the $\overline{\text{TGATE}}n$ have similar functionality, except the falling edge of $\overline{\text{TGATE}}n$ does not restart the appropriate count value in GTCNR*n*[CNV*n*].
		Timing	Assertion/Negation—Asynchronous to internal bus clock. $\overline{\text{TGATE}}n$ is internally synchronized to the system bus clock. If $\overline{\text{TGATE}}n$ meets the asynchronous input setup time, the counter begins counting after one system bus clock when working with the internal clock.
$\overline{\text{TOUT}}n$	O	Global timer counter output signal. The GTM output a signal on the timer output pin $\overline{\text{TOUT}}n$ when the reference value is reached.	
		State Meaning	Asserted/Negated—According to the programmed polarity by the corresponding GTMDR*n*[OM*n*]. 1. Active-low pulse on $\overline{\text{TOUT}}n$ for one timer input clock cycle as defined by the GTMDR*n*[ICLK*n*] bits (GTMDR*n*[OM*n*] = 1). Thus, $\overline{\text{TOUT}}n$ may be low for one general system clock period, one general system slow go clock period, or one TIN*n* pin clock cycle period. 2. Toggle the $\overline{\text{TOUT}}n$ pin (GTMDR*n*[OM*n*] = 0). $\overline{\text{TOUT}}n$ changes occur on the rising edge of the system clock.
		Timing	Assertion/Negation—$\overline{\text{TOUT}}n$ changes occur on the rising edge of the system clock.

5.7.5 GTM Memory Map/Register Definition

The GTM programmable register map occupies 64 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All GTM registers are 8 or 16 bits wide, located on 8-bit or 16-bit address boundaries, and should only be accessed as 8-bit or 16-bit quantities. All addresses used in this chapter are offsets from GPT Base, as defined in Chapter 2, “Memory Map.”

Table 5-51 shows memory map of the GTM.

Table 5-51. GTM Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
General Purpose (Global) Timer Module 1—Block Base Address 0x0_0500				
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	5.7.5.1/5-49
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	5.7.5.1/5-49
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	5.7.5.2/5-52
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0xFFFF	5.7.5.3/5-53
0x16	Timer 2 global timers reference register (GTRFR2)			
0x18	Timer 1 global timers capture register (GTCPR1)	R/W	0x0000	5.7.5.4/5-53
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	5.7.5.5/5-54
0x1E	Timer 2 global timers counter register (GTCNR2)			
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	5.7.5.2/5-52
0x22	Timer 4 global timers mode register (GTMDR4)			
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0xFFFF	5.7.5.3/5-53
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	5.7.5.4/5-53
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	5.7.5.5/5-54
0x2E	Timer 4 global timers counter register (GTCNR4)			
0x30	Timer 1 global timers event register (GTEVR1)	w1c	0x0000	5.7.5.6/5-54
0x32	Timer 2 global timers event register (GTEVR2)			
0x34	Timer 3 global timers event register (GTEVR3)			
0x36	Timer 4 global timers event register (GTEVR4)			
0x38	Timer 1 global timers prescale register (GTPSR1)	R/W	0x0003	5.7.5.7/5-55
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			
General Purpose (Global) Timer Module 2: All registers defined for GTM1 are also defined for GTM2; the base address of GTM2 registers is 0x0_06*nn*.				

5.7.5.1 Global Timers Configuration Registers (GTCFR*n*)

The global timers configuration registers (GTCFR1 and GTCFR2), shown in Figure 5-37 and Figure 5-38, contain configuration parameters used by the timers. These registers allow simultaneous starting, stopping and resetting of a pair of timers (1 and 2 or 3 and 4) or of a groups of timers (1, 2, 3, and 4) if one bus cycle is used. GTCFR is cleared by reset.

NOTE

For proper operation of the timers, do not change the modes of operation and enable the timer in the same register write operation. The modes can be changed when GTCFR*n*[RST*n*] is cleared. However, when GTCFR*n*[RST*n*] are set, they are the only bits that can be changed.

Offset 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7
R	PCAS	BCM	STP2	RST2	GM2	GM1	STP1	RST1
W								
Reset	All zeros							

Figure 5-37. Global Timers Configuration Register 1 (GTCFR1)

Table 5-52 defines the bit fields of GTCFR1.

Table 5-52. GTCFR1 Bit Settings

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer. **Note:** This bit is ignored in super-cascade mode (GTCFR2[SCAS] = 1). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1 and RST2 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	BCM	Backward compatible mode 0 Provide backward compatibility to PowerQUICC II family timers. In this mode GTCFR1[GM2] bit will control the gate mode for timers 1 and 2 and GTCFR2[GM4] bit will control the gate mode for timers 3 and 4. GTCFR1[GM1] and GTCFR2[GM3] bits are ignored. 1 Normal operational mode
2	STP2	Stop timer 2 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 2, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST2	Reset timer 2 0 Reset the timer 2, including GTMDR2, GTRFR2, GTCNR2, GTCPR2, and GTEVR2 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP2 bit is cleared.

Table 5-52. GTCFR1 Bit Settings (continued)

Bits	Name	Description
4	GM2	Gate mode for $\overline{\text{TGATE2}}$ 0 Restart gate mode. The $\overline{\text{TGATE2}}$ pin is used to enable/disable count. A low level of $\overline{\text{TGATE2}}$ enables and a falling edge of $\overline{\text{TGATE2}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level 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 appropriate count value in GTCNR2[CNV2].
5	GM1	Gate mode for $\overline{\text{TGATE1}}$ 0 Restart gate mode. The $\overline{\text{TGATE1}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE1}}$ enables and a falling edge of $\overline{\text{TGATE1}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE1}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE1}}$ does not restart the appropriate count value in GTCNR1[CNV1]. **Note:** In backward compatible mode (GTCFR1[BCM] = 0) this bit is ignored. GTCFR1[GM2] bit will control the gate mode for timers 1 and 2.
6	STP1	Stop timer 1 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 1, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	RST1	Reset timer 1 0 Reset the timer 1, including GTMDR1, GTRFR1, GTCNR1, GTCPR1, and GTEVR1 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP1 bit is cleared.

The GTCFR2 register is shown in Figure 5-38.

Offset 0x04 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	PCAS	SCAS	STP4	RST4	GM4	GM3	STP3	RST3
Reset	All zeros							

Figure 5-38. Global Timers Configuration Register 2 (GTCFR2)

Table 5-53 defines the bit fields of GTCFR2.

Table 5-53. GTCFR2 Bit Settings

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation. 1 Timers 3 and 4 cascade to form a 32-bit timer. **Note:** This bit is ignored in super-cascade mode (GTCFR2[SCAS] = 1). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST3 and RST4 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	SCAS	Super cascade mode 0 Normal operation 1 Timers 1, 2, 3 and 4 cascade to form a 64-bit timer. **Note:** In super-cascade mode (GTCFR2[SCAS] = 1) the pair-cascade mode bits are ignored, (GTCFR1/2[PCAS] = Don't Care). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1, RST2, RST3, and RST4 bits (without changing SCAS) and then, in a separate write to the register, change the value of SCAS.
2	STP4	Stop timer 4 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 4, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST4	Reset timer 4 0 Reset the timer 4, including GTMDR4, GTRFR4, GTCNR4, GTCPR4, and GTEVR4 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP4 bit is cleared.
4	GM4	Gate mode for $\overline{\text{TGATE4}}$ 0 Restart gate mode. The $\overline{\text{TGATE4}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE4}}$ enables and a falling edge of $\overline{\text{TGATE4}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE4}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE4}}$ does not restart the appropriate count value in GTCNR4[CNV4].
5	GM3	Gate mode for $\overline{\text{TGATE3}}$ 0 Restart gate mode. The $\overline{\text{TGATE3}}$ is used to enable/disable count. A low level of $\overline{\text{TGATE3}}$ enables and a falling edge of $\overline{\text{TGATE3}}$ restarts the count (reset the dynamic counter's count value to 0) and a high level of $\overline{\text{TGATE3}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE3}}$ does not restart the appropriate count value in GTCNR3[CNV3]. **Note:** In backward compatible mode (GTCFR1[BCM] = 0) this bit is ignored. The GTCFR2[GM4] bit controls the gate mode for timers 3 and 4.
6	STP3	Stop timer 3 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 3, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	RST3	Reset timer 3 0 Reset the timer 3, including GTMDR3, GTRFR3, GTCNR3, GTCPR3, and GTEVR3 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP3 bit is cleared.

5.7.5.2 Global Timers Mode Registers (GTMDR1–GTMDR4)

The global timers mode registers (GTMDR1, GTMDR2, GTMDR3, and GTMDR4) are shown in Figure 5-39.

Erratic behavior may occur if GTCFR1 and GTCFR2 are not initialized before the GTMDR*n*. Only GTCFR*n*[RST*n*] and GTCFR*n*[STP*n*] can be modified at any time.

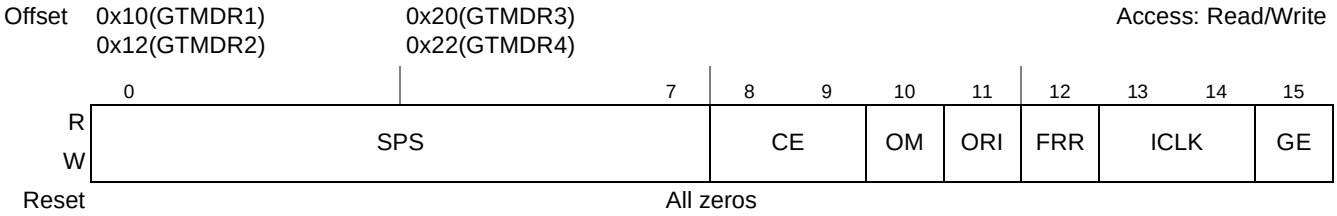


Figure 5-39. Global Timers Mode Registers (GTMDR1–GTMDR4)

Table 5-54 defines the bit fields of GTMDR.

Table 5-54. GTMDR Bit Settings

Bits	Name	Description
0–7	SPS	Secondary prescaler value The secondary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.
8–9	CE	Capture edge and enable interrupt 00 Disable interrupt on capture event; capture function is disabled 01 Capture on rising TIN*n* edge only and enable interrupt on capture event. 10 Capture on falling TIN*n* edge only and enable interrupt on capture event. 11 Capture on any TIN*n* edge and enable interrupt on capture event. **Note:** The frequency of TIN*n* should be slower than system clock (TIN*n* is sampled internally by system clock to detect TIN*n*'s rising/falling edge before updating the counter)
10	OM	Output mode 0 Toggle $\overline{\text{TOUT}n}$ every time when the corresponding timer matches its reference value. 1 Active-low pulse on $\overline{\text{TOUT}n}$ for one timer input clock cycle (4 input clock cycles for the system clock) as defined by the ICLK*n* bits. Thus, $\overline{\text{TOUT}n}$ may be low for four general system clocks, one general system slow go clock period, or one TIN*n* pin clock cycle period. **Note:** $\overline{\text{TOUT}n}$ changes are internally synchronized to the rising edge of the system clock
11	ORI	Output reference interrupt enable 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt on reaching the reference value.
12	FRR	Free run/restart mode 0 Free run. The timer count continues to increment after the reference value is reached. 1 Restart. The timer count is reset immediately after the reference value is reached.

Table 5-54. GTMDR Bit Settings (continued)

Bits	Name	Description
13–14	ICLK	Input clock source for the timer. 00 Internally cascaded input. This selection means: For ICLK1, the timer 1 input is the output of timer 2. For ICLK2, the timer 1 input is the output of timer 2, the timer 2 input is the output of timer 3, the timer 3 input is the output of timer 4. For ICLK3, the timer 3 input is the output of timer 4. For ICLK4 this selection means no input clock is provided to the timer. 01 Internal general system bus clock. 10 Internal slow go clock (divided by 16 system bus clock). 11 TIN*n*: corresponding TIN1, TIN2, TIN3, or TIN4 pin (falling edge).
15	GE	Gate enable 0 The $\overline{\text{TGATE}n}$ signal is ignored. 1 The $\overline{\text{TGATE}n}$ signal is used to control the timer.

5.7.5.3 Global Timers Reference Registers (GTRFR1–GTRFR4)

Global timers reference registers, shown in Figure 5-40, are 16-bit memory-mapped, read/write registers containing the 16-bit reference values for each timer's timeout. The reference value is not reached until GTCNR*n*[CNV] increments to the value in GTRFR*n*[TRV].

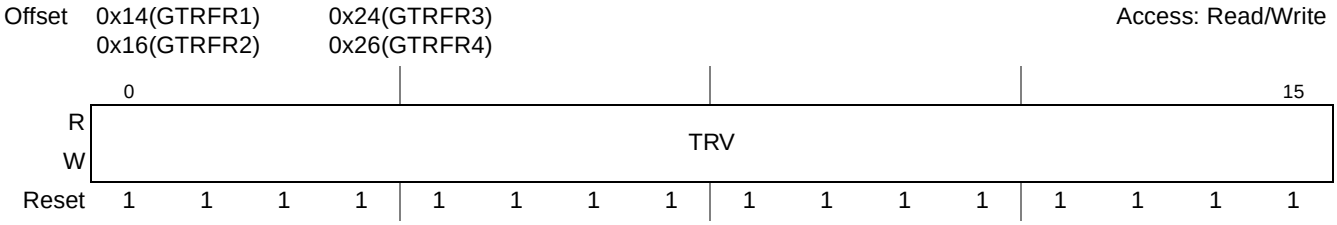


Figure 5-40. Global Timers Reference Registers (GTRFR1–GTRFR4)

Table 5-55 defines the bit fields of GTRFR.

Table 5-55. GTRFR Bit Settings

Bits	Name	Description
0–15	TRV	Timeout reference value. 16-bit timeout reference value for the corresponding timer. Set to all ones by reset.

5.7.5.4 Global Timers Capture Registers (GTCPR1–GTCPR4)

Global timers capture registers (GTCPR1, GTCPR2, GTCPR3, and GTCPR4), shown in Figure 5-41, are used to latch the value of the counters according to GTMDR*n*[CE].

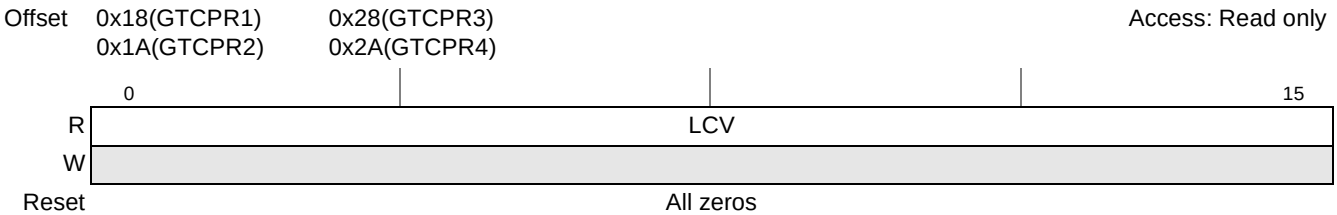


Figure 5-41. Global Timers Capture Registers (GTCPR1–GTCPR4)

Table 5-56 defines the bit fields of GTCPR*n*.

Table 5-56. GTCPR*n* Bit Settings

Bits	Name	Description
0–15	LCV	Latched counter value. Corresponding timer's 16-bit latched value.

5.7.5.5 Global Timers Counter Registers (GTCNR1–GTCNR4)

Global timers counter registers (GTCNR1, GTCNR2, GTCNR3, and GTCNR4), shown in Figure 5-42, are four 16-bit, memory-mapped, read/write up-counters. A read cycle to a GTCNR*n*[CNV] fields yields the current value of the appropriate timer but does not affect the counting operation. A write cycle to a GTCNR*n*[CNV] field sets the register to the written value, causing its corresponding primary and secondary prescaler counters to be reset.

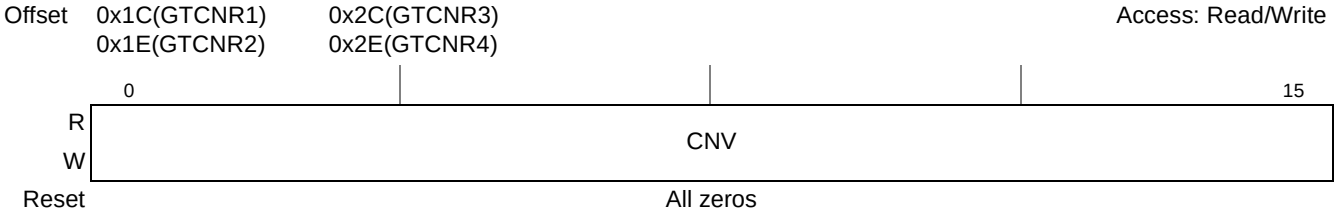


Figure 5-42. Global Timers Counter Registers (GTCNR1—GTCNR4)

Table 5-57 defines the bit fields of GTCNR.

Table 5-57. GTCNR Bit Settings

Bits	Name	Description
0–15	CNV	Counter value. Corresponding timer's 16-bit read/write up-counter value.

5.7.5.6 Global Timers Event Registers (GTEVR1–GTEVR4)

Global timers event registers (GTEVR1, GTEVR2, GTEVR3, and GTEVR4), shown in Figure 5-43, are used to report events recognized by any of the timers. On recognition of an output reference event, the appropriate timer sets GTEVR*n*[REF], regardless of the corresponding GTMDR*n*[ORI]. The capture event is only set if it is enabled by GTMDR*n*[CE]. GTEVRs appear as memory-mapped registers to users, which can be read at any time.

GTEVR*n* bits are cleared by writing ones to them (writing zeros does not affect bit values). Both bits must be reset before the timer negates the interrupt to the interrupt controller.

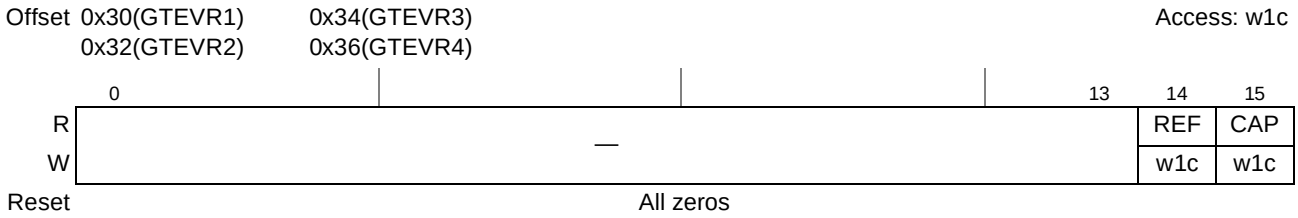


Figure 5-43. Global Timers Event Registers (GTEVR1—GTEVR4)

Table 5-58 defines the bit fields of GTEVR*n*.

Table 5-58. GTEVR*n* Bit Settings

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	REF	Output reference event 0 No event 1 The counter reached the GTRFR*n*[TRV] value. GTMDR*n*[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Counter capture event Corresponding timer's 16-bit read/write up-counter value. 0 No event 1 The counter value has been latched into the GTCPR*n*[LCV]. GTMDR*n*[CE] is used to enable generation of this event.

5.7.5.7 Global Timers Prescale Registers (GTPSR1–GTPSR4)

The global timers prescale registers (GTPSR1, GTPSR2, GTPSR3, and GTPSR4) are shown in Figure 5-44.

Erratic behavior may occur if GTPSR*n* is not initialized before the corresponding GTMDR*n*.

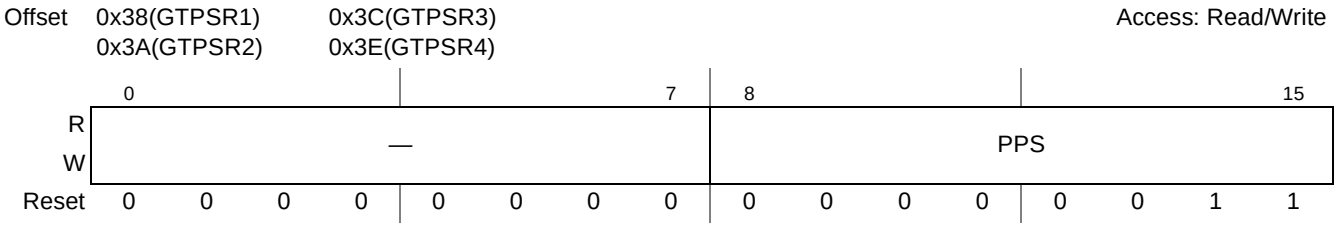


Figure 5-44. Global Timers Prescale Registers (GTPSR1–GTPSR4)

Table 5-59 defines the bit fields of GTPSR*n*.

Table 5-59. GTPSR*n* Bit Settings

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	PPS	Primary prescaler bits The primary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.

NOTE

The total timer prescale value is calculated as follows:

$$GTMn_{prescaler} = (GTPSRn[PPS] + 1) \cdot (GTMDRn[SPS] + 1)$$

This gives a total prescale range from 1 (GTPSR*n*[PPS] = 0x00, GTMDR*n*[SPS] = 0x00) to 65,536 (GTPSR*n*[PPS] = 0xFF, GTMDR[SPS] = 0xFF).

5.7.6 Functional Description

5.7.6.1 General-Purpose Timer Units

The clock input to the timer's prescaler can be selected from the following sources:

- The system clock
- The system slow go clock (internally divided by 16)
- The corresponding TIN*n* signal, usually programmed in the general purpose I/O (GPIO) registers

The general system clock is generated in the clock synthesizer and defaults to the system frequency. However, the general system clock has the option to be divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the general system clock, the user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer. Alternatively, the user may prefer TIN*n* to be the clock source. TIN*n* is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding GTMDR*n*[ICLK] bits. The prescalers (GTMDR*n*[SPS] and GTPSR*n*[PPS]) can be programmed to divide the clock input by values from 1 to 65,537 and the output of the prescaler is used as an input to the 16-bit counters. The best resolution of the timer is one clock cycle (3 ns at a 333-MHz system clock, for example). The maximum period (when the reference value is all ones and the prescaler divides by 256) for one 16-bit timer is ~50 ms at 333 MHz.

5.7.6.2 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding GTMRR selects each mode.

- Free run reference mode (GTMDR*n*[FRR] = 0)

  The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode (GTMDR*n*[FRR] = 1)

  The corresponding timer count is reset immediately after the reference value is reached.

Upon reaching the reference value, the corresponding GTEVR*n*[REF] bit is set and an interrupt is issued if GTMDR*n*[ORI] = 1. The timers can output a signal on the timer output pin $\overline{\text{TOUT}n}$ if the reference value is reached (selected by the corresponding GTMDR*n*[OM]). This signal can be an active-low pulse or a toggle of the current output. The output can also be connected internally to the input of another timer, resulting in a 32- or 64-bit timer.

5.7.6.3 Capture Modes

In addition, each timer has a 16-bit field in GTCPR, used to latch the value of the counter when a defined transition of TIN*n* is sensed by the corresponding input capture edge detector. The timers may be gated/restarted by an external gate signals ($\overline{\text{TGATE}n}$) that controls the timers. The type of transition triggering the capture is selected by the corresponding GTMDR*n*[CE] bits. Upon a capture or reference

event, corresponding GTEVR*n*[REF] or GTEVR*n*[CAP] is set and a maskable interrupt request is issued to the interrupt controller.

- Normal gate mode enables the count on a falling edge of $\overline{\text{TGATE}}$ and disables the count on the rising edge of $\overline{\text{TGATE}}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}}$.
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of $\overline{\text{TGATE}}$.

  This mode has applications in pulse interval measurement and bus monitoring as follows:

  — Pulse measurement—The restart gate mode can measure a low pulse on $\overline{\text{TGATE}}$. The rising edge of $\overline{\text{TGATE}}$ completes the measurement and if $\overline{\text{TGATE}}$*n* is connected externally to TIN*n*, it causes the timer to capture the count value and generate a rising-edge interrupt.

  — Bus monitoring—The restart gate mode can detect a signal that is stuck abnormally low. The bus signal should be connected to $\overline{\text{TGATE}}$. The timer count is reset on the falling edge of the bus signal and if the bus signal does not go high again within the number of user-defined clocks, an interrupt can be generated.

The gate function is enabled in the GTMDR; the gate operating mode is selected in the GTCFR*n*.

NOTE

$\overline{\text{TGATE}}$ is internally synchronized to the system clock. If $\overline{\text{TGATE}}$ meets the asynchronous input setup time, the counter begins counting after one system clock when working with the internal clock.

5.7.6.4 Cascaded Modes

GTCFR*n*[PCAS] and GTCFR2[SCAS] are used to put the timers into different cascaded modes:

- Non-cascaded mode (GTCFR*n*[PCAS] = 0 and GTGCF2[SCAS] = 0)

  If GTCFR*n*[PCAS] = 0 and GTCFR2[SCAS] = 0, the each timer (timer 1, timer 2, timer 3, and timer 4), function as a independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR, and GTCNR for each one (Figure 5-45). When working in the none-cascaded mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with appropriate 16-bit bus cycles.

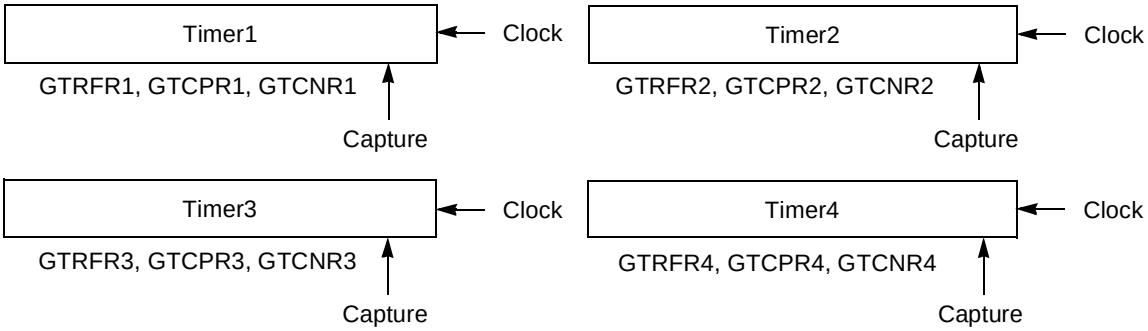


Figure 5-45. Timers Non-Cascaded Mode Block Diagram

- Pair-cascaded mode (GTCFR1[PCAS] = 1 and/or GTCFR2[PCAS] = 1, GTCFR2[SCAS] = 0)

  In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 may be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4, as shown in Figure 5-46. Because the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer (GTCFR1[PCAS] = 1, GTCFR2[PCAS] = 0 or GTCFR1[PCAS1] = 0, GTCFR2[PCAS] = 1), or two 32-bit timers (GTCFR1[PCAS] = 1 and GTCFR2[PCAS] = 1).

  If GTCFR1[PCAS] = 1 and/or GTCFR2[SCAS] = 1, the two 16-bit timers (timer 1 and timer 2 or timer 3 and timer 4) function as a 32-bit timer with a 32-bit GTRFR, GTCPR, and GTCNR. In this case, GTMDR1/GTMDR3 is ignored, and the modes and functions are defined using GTMDR2/GTMDR4, and GTCFR1/GTCFR2. The capture are controlled from TIN2, and the interrupts are generated from GTEVR2. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.

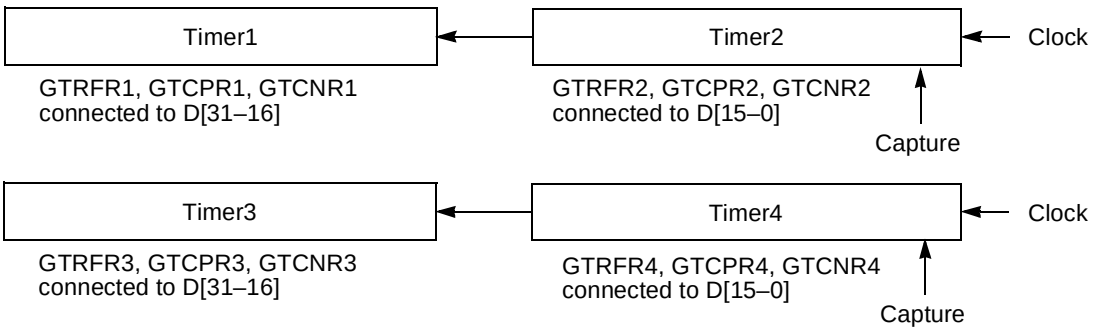


Figure 5-46. Timer Pair-Cascaded Mode Block Diagram

- Super-cascaded mode (GTCFR2[SCAS] = 1)

  In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter, as shown in Figure 5-47.

  If GTCFR2[SCAS] = 1, the all four 16-bit timers function as a 64-bit timer with a cascaded 32-bit GTRFR, GTCPR, and GTCNR. In this case, registers GTMDR1, GTMDR2, GTMDR3, and GTCFR1 are ignored, and the modes and functions are defined using GTMDR4 and GTCFR2 only. The capture are controlled from TIN4, and the interrupts are generated from GTEVR4. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

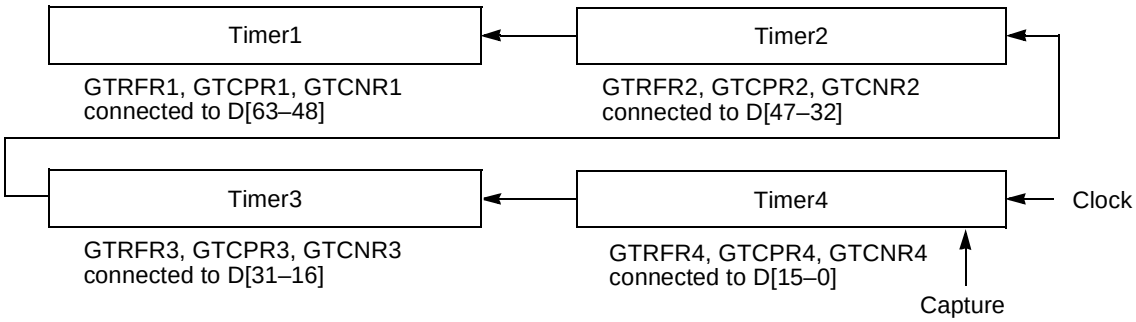


Figure 5-47. Timers Super-Cascaded Mode Block Diagram

5.7.7 Initialization/Application Information

5.7.7.1 Programming Guidelines

5.7.7.1.1 GTM Registers

The following initialization sequence of GTM is recommended:

- Write to GTCFR*n* in order to reset, to stop or to configure the appropriate timer's operation: cascaded timers configuration, gate mode configuration.
- Write to GTPSR*n*[PPS] fields in order to program the appropriate timer's clock primary prescaler.
- Write to GTMDR*n* in order to choose an input clock, to program the secondary prescaler and to set a desirable appropriate timer's operational mode.

NOTE

Erratic behavior may occur if GTCFR*n* and GTPSR are not initialized before the GTMDR. Only GTCFR*n*[RST*n*] can be modified at any time

- Clear GTEVR*n*[REF] and GTEVR*n*[CAP] by writing 1s in order to clear the previous events.
- Write to GTRFR and to GTCNR*n* according to appropriate timer's GTMDR*n* programming.

NOTE

A write cycle to a GTCNR*n*[CNV] fields sets the register to the written value, causing its corresponding primary and secondary prescalers, (GTPSR*n*[PPS] and GTMDR*n*[SPS]), to be reset.

- Write to GTCFR*n*[STP*n*] and to GTCFR*n*[RST*n*] in order to initialize the appropriate timer's operation.

5.8 Power Management Control (PMC)

The device provides a power management control (PMC) unit, which enables the device to smoothly enter and exit low power modes. Low power modes may be used when internal units in the device temporarily or permanently do not perform any action.

The device uses one or more of the following methods for power saving:

- Dynamic power management
- Shutting down unused blocks
- Software-controlled power-down states

5.8.1 External Signal Description

Table 5-60 describes the power management signals.

Table 5-60. System Control Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{QUIESCE}}$	O	Quiesce state. Indicates that the processor system and PowerPC core are in low power state.	
		State Meaning	Asserted—The system and PowerPC core are in low power state. Negated—The system and PowerPC core are not in low power state.
		Timing	The timing between a quiesce request from the PowerPC core and the assertion of the external indication or between negation of the core's quiesce request and negation of the external indication depends on the current state of the internal system units and may vary accordingly.

5.8.2 PMC Memory Map/Register Definition

Table 5-61 shows the memory map for the power management controller registers.

Table 5-61. Power Management Controller Registers Memory Map

Offset	Register	Access	Reset	Section/Page
0x00B00	Power management controller configuration register (PMCCR)	R/W	0x0000_0000	5.8.2.1/5-60
0x00B04	Power management controller event register (PMCER)	R/W	0x0000_0000	5.8.2.2/5-61
0x00B08	Power management controller mask register (PMCMR)	R/W	0x0000_0000	5.8.2.3/5-62
0x00B0C–0x00BFC	Reserved	—	—	—

5.8.2.1 Power Management Controller Configuration Register (PMCCR)

The power management controller configuration register (PMCCR), shown in Figure 5-48, controls whether only the PowerPC core will enter low power state upon quiesce request or additional parts of the device will also enter low power state.

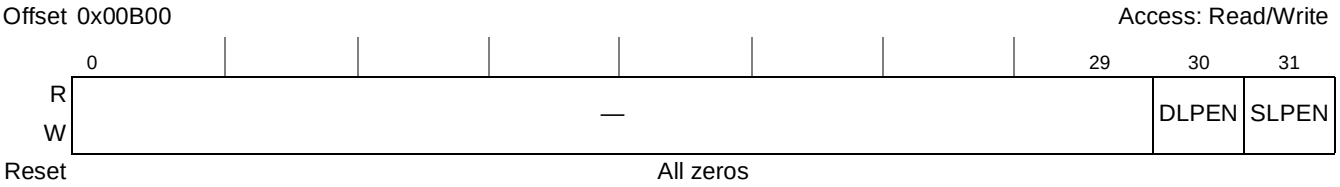


Figure 5-48. Power Management Controller Configuration Register

Table 5-5 defines the bit fields of PMCCR.

Table 5-62. PMCCR Bit Settings

Bits	Name	Description
0–29	—	Reserved. Write has no effect, read returns 0.
30	DLPEN	DDR SDRAM low power enable 0 The DDR SDRAM memory controller is prevented from entering low power state. 1 The DDR SDRAM memory controller will enter low power state when the rest of the system enters low power, according to SLPEN setting. DDR SDRAM will enter self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) and DDR clocks (MCK*n*) are shut off. This bit is cleared when the device exits from low power state. Note that setting this bit without setting SLPEN has no effect.
31	SLPEN	System low power enable 0 The system is prevented from entering low power state. 1 The system will enter low power state when a quiesce request from the PowerPC core arrives. This bit is cleared when the device exits from low power state.

5.8.2.2 Power Management Controller Event Register (PMCER)

The power management controller event register (PMCER), shown in Figure 5-49, indicates with the PMCI bit that the power management controller has detected a wake-up event, that the system is not in idle state anymore, and that the device should exit low power state. If PMCMR[PMCIE] is set, the PMC interrupt request to the PowerPC core is driven.

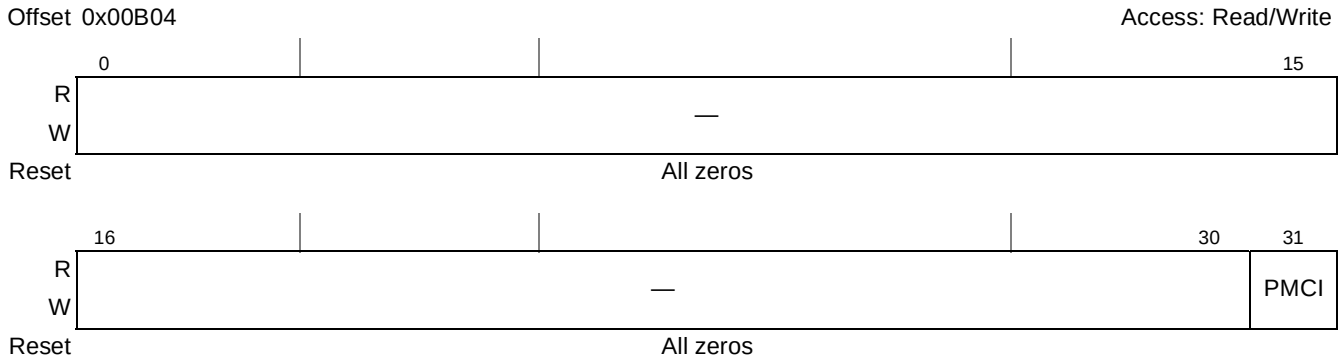


Figure 5-49. Power Management Controller Event Register

Table 5-63 defines the bit fields of PMCER.

Table 5-63. PMCER Bit Settings

Bits	Name	Description
0–30	—	Reserved. Write has no effect, read returns 0.
31	PMCI	Power management controller interrupt. When set, indicates that the power management controller has detected that the system is not in idle state anymore, and that the device is required to exit low power state. If PMCMR[PMCIE] is set, the PMC interrupt request to the PowerPC core is driven, causing the PowerPC core to exit its low power state. PMCI can be cleared by writing a 1 to it (writing zero has no effect).

5.8.2.3 Power Management Controller Mask Register (PMCMR)

The power management controller mask register (PMCMR), shown in Figure 5-50, controls through the PMCIE bit whether the PMC interrupt request to the PowerPC core is enabled. The PMC interrupt request causes the PowerPC core to exit its low power state before any transaction on the system bus occurs.

Figure 5-50. Power Management Controller Mask Register

Table 5-64 defines the bit fields of PMCMR.

Table 5-64. PMCMR Bit Settings

Bits	Name	Description
0–30	—	Reserved. Write has no effect, read returns 0.
31	PMCIE	Power management controller interrupt enable. 0 PMC interrupt request (PMCI) is disabled. 1 PMC interrupt request (PMCI) is enabled.

NOTE

The user is also required to enable the PMC interrupt in the programmable interrupt controller by setting SIMSR_L[PMC].

5.8.3 Functional Description

The device has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Software can also shut down clocks to individual blocks when they are not needed through a memory-mapped register in the clock unit (SCCR). Additionally, software running on the PowerPC core can access the core's SPRs to put the device into doze, nap, or sleep power down states. Finally, software can access the PMCCR register to enable the device to go to low power state whenever the PowerPC core enters nap or sleep states. These power management features are described in further detail in this section.

5.8.3.1 Dynamic Power Management

Many blocks in the device can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

5.8.3.2 Shutting Down Unused Blocks

As described in Section 4.5.2.3, "System Clock Control Register (SCCR)," SCCR provides a way to shut down certain functional blocks within the device when they are not needed in a particular system. SCCR can be written by the PowerPC core or by an external master. Powering down a block in this way turns off all clocks to that block. It does not remove power. It is required that the SCCR is written to shut down a certain functional block only when that block is idle.

NOTE

Functional blocks disabled using SCCR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

5.8.3.3 Software-Controlled Power-Down States

PowerPC software can place the core in doze, nap, or sleep power-down states by writing to HID0 in the core, as described in detail in the section "Hardware Implementation Register 0 (HID0)," of the *e300 PowerPC Core Reference Manual*. In addition, if PMCCR[SLPEN] is set when the PowerPC core request to enter nap or sleep modes, it will also cause the system internal logic units to enter low power mode.

5.8.3.3.1 Entering Low Power States—Core-Only Mode

Entering Doze mode is controlled only by the e300 PowerPC core itself, and does not involve the power management controller or other blocks.

Entering Nap or Sleep modes occurs by writing to HID0 in the core, causing the core to make a quiesce request to the power management controller while PMCCR[SLPEN] is cleared. The core is immediately enabled to enter low power state, regardless of the system status. Note that because the core does not snoop the bus in this mode, it is the user's responsibility to keep the cache coherent. Other device peripheral and internal units continue to operate in full-on mode while the core is in low power state in this mode.

5.8.3.3.2 Entering Low Power States—Core and System Mode

Core and system mode is achieved when the core makes a quiesce request to the power management controller after PMCCR[SLPEN] is set. To preserve cache coherency and otherwise avoid loss of system state, the core's transition to low-power modes is coordinated with other functional blocks. The power management controller allows the core to enter power down mode only when the rest of the system is idle.

When the power management controller detects that the internal system bus is idle, and there are no outstanding transactions, it signals the internal logic units to enter low power state.

If PMCCR[DLPEN] is set, the DDR SDRAM is first set to self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) before the memory controller stops driving refresh commands. Self-refresh mode guarantees that the memory content will remain valid while the memory controller and its clocks are off. The DLL is then shut off and stops driving clocks on MCK*n* pins. Finally the DDR SDRAM memory controller enters low power state and acknowledges the power management controller.

The power management controller then signals the core and acknowledges its request to enter power down mode. Finally the $\overline{\text{QUIESCE}}$ output signal is asserted.

5.8.3.4 Exiting Core and System Low Power States

The device can exit low power state and return to full-on mode for one of the following reasons:

- The core internal time base unit invokes a request to exit low power state.
- The core has received an interrupt request.
- The power management controller has detected that the system is not idle and there are outstanding requests for transactions on the internal bus.

The actions taken to exit low power state depend on the mode and whether the system or only the core are in this state. The following sections describe the various scenarios.

5.8.3.4.1 Exiting Low Power States—Core-Only Mode

Exit from Doze mode is controlled only by the core itself and does not involve the power management controller or other blocks in the device.

Nap or Sleep modes are exited when the core has received an interrupt request, or according to the internal time base unit of the core (Nap mode only). The source of the interrupt can be an internal block or external signal. When the core returns to full-on state, it signals to the power management controller that it is ready and is immediately acknowledged to access the rest of the system.

5.8.3.4.2 Exiting Low Power States—Core and System Mode

The power management controller decides to exit low power state when it detects that the system is not idle anymore. The device may exit idle state when one of the peripheral interfaces makes a request to access the internal bus or when the core returns to full-on state, as described above, and makes a request to access the internal bus. For example, the QUICC Engine block receives an Ethernet frame, and requires to store it on the DDR SDRAM memory.

If the DDR SDRAM memory controller is in low power state (PMCCR[DxLPEN] was set when entering low power state), the power management controller initially enables the DDR SDRAM memory controller and its DLL, allowing it to lock. When locked, DDR SDRAM clocks (MCK*n)* are enabled and the memory controller exit self-refresh and returns to auto-refresh mode.

The power management controller then enables other internal units and interrupts the PowerPC core. When all internal units, including the core, are ready, the power management controller enables the device to return to full-on state, negate the $\overline{\text{QUIESCE}}$ output, and clear PMCCR[SLPEN]. Outstanding requests for transactions are now granted to execute on the internal bus.

NOTE

Software is required to enable PMCI interrupt by setting PMCMR[PMCIE], otherwise exiting from low power state is not possible.

NOTE

It is the software’s responsibility to clear PMCER[PMCI].

5.8.4 Initialization/Application Information

5.8.4.1 Core Disable in Low Power Mode

If the device is required to operate with the core permanently disabled, the following steps must be taken:

1. Initialize the device with the core enable.
2. Clear PMCCR[SLPEN] and disable the core time base unit by clearing SPCR[TBEN]. See Section 5.3.2.4, “System Priority and Configuration Register (SPCR),” for more information.
3. The e300 core enters low power state by accessing the HID0 register.
4. Set ACR[COREDIS] in the system arbiter register with an external master (that is, PCI). This steers all device interrupts to the $\overline{\text{PCI\_INTA}}$ so the core cannot receive any interrupt requests.

NOTE

Make sure that the core cannot receive any interrupt requests during the time interval between setting HID0 and setting ACR[COREDIS]. This can be achieved if the rest of the system is idle (during the initialization).

By following this flow, the e300 core remains in low power state while the rest of the system is operational, and does not get out of this state as a result of any interrupt or time-based event.

Chapter 6
Arbiter and Bus Monitor

This chapter describes operation theory of the arbiter in the device. In addition, it describes configuration, control, and status registers of the arbiter.

6.1 Overview

The arbiter is responsible for providing coherent system bus arbitration. It tracks all address and data tenures and provides all the arbitration signals to masters and slaves. In addition, it monitors the bus and reports on errors and protocol violations.

The arbiter includes the following features:

- Supports a programmable pipeline depth (from 1 to 4)
- Supports four levels of priority for bus arbitration
- Supports repeat-request mode: number of programmable consecutive transactions from the same master (up to eight transactions)
- Supports data streaming operations
- Supports programmable address bus parking mode: disable, park to last bus owner, park to software selected master
- Claims address only, reserved and illegal transaction types, report on it and can raise maskable interrupt
- Provides timers for address tenure time out and data tenure time out detection and can issue maskable interrupt, if any timer expired
- Reports on transfer error and can issue maskable interrupt
- Can issue regular or machine check interrupt for each type of error event (programmable)

6.1.1 Coherent System Bus Overview

The coherent system bus is the central bus of the device. Any data transaction from master to slave in the device passes through the coherent system bus. The device coherent system bus supports pipelined transactions. It has independent address and data tenures. Pipeline depth determines the number of address tenures that can be started before the first data tenure is finished.

Basic burst size is equal to cache line length of the core, which is 32 bytes. Using repeat request mode enables up to eight consecutive bursts to be executed by the same master. Maximum number of consecutive transactions can be limited by programming arbiter configuration register. See Section 6.2.1, “Arbiter Configuration Register (ACR),” for more details.

NOTE

Write accesses to different interfaces are not guaranteed to finish in order.

6.2 Arbiter Memory Map/Register Definition

Table 6-1 shows the memory map for arbiter's configuration, control and status registers.

Table 6-1. Arbiter Register Map

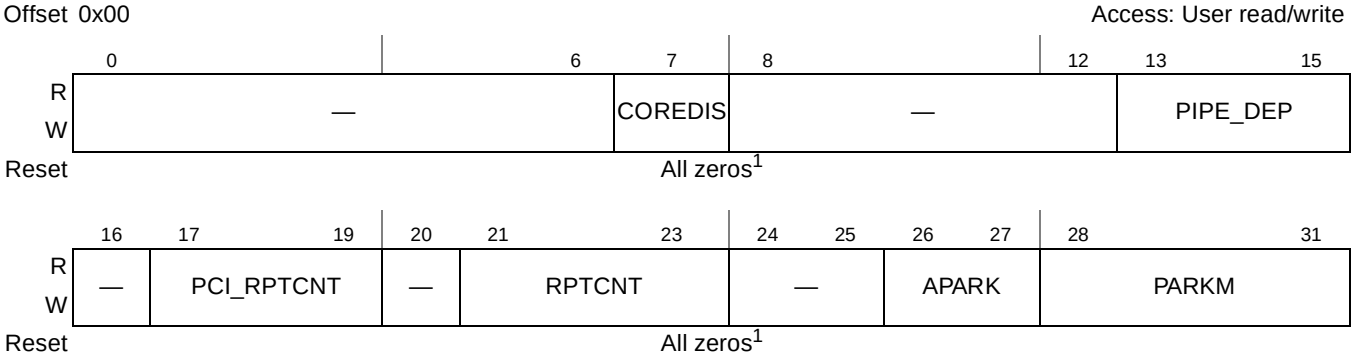
Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00	Arbiter configuration register (ACR)	R/W	0x0000_0000/ 0x0010_0000[1]	6.2.1/6-2
0x04	Arbiter timers register (ATR)	R/W	0x00FF_00FF	6.2.2/6-4
0x0C	Arbiter event register (AER)	w1c	0x0000_0000	6.2.3/6-5
0x10	Arbiter interrupt definition register (AIDR)	R/W	0x0000_0000	6.2.4/6-6
0x14	Arbiter mask register (AMR)	R/W	0x0000_0000	6.2.5/6-7
0x18	Arbiter event attributes register (AEATR)	R	0x0000_0000[2]	6.2.6/6-7
0x1C	Arbiter event address register (AEADR)	R	0x0000_0000[2]	6.2.7/6-9
0x20	Arbiter event response register (AERR)	R/W	0x0000_0000	6.2.8/6-10

1 Reset value is determined from the core PLL configuration of the reset configuration word. See Chapter 4, "Reset, Clocking, and Initialization," for details.

2 The registers AEATR and AEADR are affected only by the assertion of $\overline{\text{PORESET}}$

6.2.1 Arbiter Configuration Register (ACR)

Arbiter configuration register (ACR) defines the arbiter modes and parked master on the bus. Figure 6-1 shows the fields of ACR.



1 Note that the reset value of COREDIS and bits 10–11 are determined from reset configuration word. (See Section 4.3.2, "Reset Configuration Words," for more details on reset configuration word.)

Figure 6-1. Arbiter Configuration Register (ACR)

Table 6-2 describes ACR fields.

Table 6-2. ACR Field Descriptions

Bits	Name	Description
0–6	—	Write reserved, read = 0
7	COREDIS	Core disable. Specifies whether CPU is disabled. When CPU is disabled, it cannot be granted on the bus by the arbiter. After reset, this bit receives its value from the reset configuration bit of COREDIS and can be configured by software. Also, if boot source is boot sequencer, COREDIS must be set to 1 at reset and the last transaction of the boot sequencer must set COREDIS to 0, if CPU enable is needed. 0 CPU enabled. 1 CPU disabled.
8–9	—	Write reserved, read = 0
10–11	—	Reserved. Write should preserve reset value.The reset value is a function of the core PLL configuration, which is part of the reset configuration word. When the core is set to operate at 1:1 or 3:2 bus clock, these bits are set to ‘01’ during reset; otherwise, they are set to ‘00’.
12	—	Write reserved, read = 0
13–15	PIPE_DEP	Pipeline depth (number of outstanding transactions). 000 Pipeline depth 1 (1 outstanding transaction) 001 Pipeline depth 2 (2 outstanding transactions) 010 Pipeline depth 3 (3 outstanding transactions) 011 Pipeline depth 4 (4 outstanding transactions) 1xx Reserved
16	—	Write reserved, read = 0
17–19	PCI_RPTCNT	PCI repeat count. Specifies the maximum number of consecutive transactions, that PCI master can perform, using $\overline{\text{REPEAT}}$ request mode. 000 One consecutive transaction ($\overline{\text{REPEAT}}$ request mode disable) 001 Two consecutive transactions 010 Three consecutive transactions 011 Four consecutive transactions 100 Five consecutive transactions 101 Six consecutive transactions 110 Seven consecutive transactions 111 Eight consecutive transactions
20	—	Write reserved, read = 0
21–23	RPTCNT	Repeat count. Specifies the maximum number of consecutive transactions, that any master (except PCI) can perform, using $\overline{\text{REPEAT}}$ request mode. 000 1 consecutive transactions ($\overline{\text{REPEAT}}$ request mode disable) 001 2 consecutive transactions 010 3 consecutive transactions 011 4 consecutive transactions 100 5 consecutive transactions 101 6 consecutive transactions 110 7 consecutive transactions 111 8 consecutive transactions **Note:** It is recommended not to program this field for more than four consecutive transactions.
24–25	—	Write reserved, read = 0

Table 6-2. ACR Field Descriptions (continued)

Bits	Name	Description
26–27	APARK	Address parking. Specifies arbiter bus parking mode. 00 Park to master. Arbiter parks the address bus to the master, that is selected by numeric value of PARKM field. 01 Park to last owner. Arbiter parks the address bus to last bus owner. 10 Disable. Arbiter does not assert $\overline{BG}$ to any master, if no $\overline{BR}$ is present. 11 Reserved
28–31	PARKM	Parking master. 0000 e300 core 0001 PCI, DMA 0010 Reserved 0011 QUICC Engine block 0100 Encryption core 0101–1111Reserved

6.2.2 Arbiter Timers Register (ATR)

The arbiter timers register (ATR) defines the arbiter address time out (ATO) and data time out (DTO) values. Figure 6-2 shows the fields of ATR.

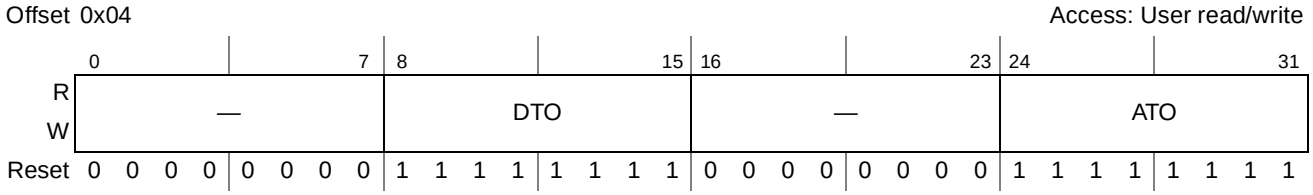


Figure 6-2. Arbiter Timers Register (ATR)

Table 6-3 describes ATR fields.

Table 6-3. ATR Field Descriptions

Bits	Name	Description
0–7	—	Write reserved, read = 0
8–15	DTO	Data time out. Specifies the time-out period for the data tenure. The granularity of this field is64128 bus clocks. The maximum value is163208355840 coherent system bus clocks. Data time_out occurs if the data tenure does not end before the specified time-out period. When DTO = n, the timeout cycle is n*64. 0x00 Reserved 0x01 64 clock cycles 0x10 128 clock cycles 0x11 192 clock cycles ... 0xFF 16320 clock cycles

Table 6-3. ATR Field Descriptions (continued)

Bits	Name	Description
16–23	—	Write reserved, read = 0
24–31	ATO	Address time out. Specifies the time-out period for the address tenure. The granularity of this field is64128 bus clocks. Maximum value is163208355840 coherent system bus clocks. Address time-out occurs if the address tenure did not end before the specified time-out period. When ATO = n, the timeout cycle is n*64. 0x00 Reserved 0x01 64 clock cycles 0x10 128 clock cycles 0x11 192 clock cycles ... 0xFF 16320 clock cycles

6.2.3 Arbiter Event Register (AER)

The arbiter uses arbiter event register (AER) to report on erroneous transactions. This register is cleared by writing ones to the fields to be cleared. Figure 6-3 shows the fields of AER.

Offset 0x0C Access: User w1c

	0 – 25	26	27	28	29	30	31
R	—	ETEA	RES	ECW	AO	DTO	ATO
W		w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros						

Figure 6-3. Arbiter Event Register (AER)

Table 6-4 describes AER fields.

Table 6-4. AER Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error.Reports on detection of transfer error by one of the slaves. 0 No transfer error detected by one of the slaves. 1 Transfer error detected by one of the slaves.
27	RES	Reserved transfer type. Reports on transaction with reserved transfer type. See Section 6.3.2.5, “Reserved Transaction Type,” for more information. 0 No transaction with reserved transfer type occurred. 1 Transaction with reserved transfer type occurred.
28	ECW	External control word transfer type. Reports on transaction with external control word transfer type. See Section 6.3.2.6, “Illegal (eciwx/ecowx) Transaction Type,” for more information. 0 No transaction with external control word transfer type occurred. 1 Transaction with external control word transfer type occurred.
29	AO	Address Only transfer type. Reports on transaction with address only transfer type. See Section 6.3.2.4, “Address Only Transaction Type,” for more information. 0 No transaction with address only transfer type occurred. 1 Transaction with address only transfer type occurred.

Table 6-4. AER Field Descriptions (continued)

Bits	Name	Description
30	DTO	Data time out. Reports on data tenure time out. 0 Data time out timer is not expired. 1 Data time out timer is expired.
31	ATO	Address time out. Reports on address tenure time out. 0 Address time out timer is not expired. 1 Address time out timer is expired.

6.2.4 Arbiter Interrupt Definition Register (AIDR)

Arbiter interrupt definition register (AIDR) determines the interrupt that responds to different error conditions. Setting a bit defines the corresponding interrupt as MCP interrupt; clearing a bit defines the corresponding interrupt as regular interrupt. Figure 6-4 shows the fields of AIDR.

Offset 0x10 Access: User read/write

	0	25	26	27	28	29	30	31
R W	—		ETEA	RES	ECW	AO	DTO	ATO
Reset	All zeros							

Figure 6-4. Arbiter Interrupt Definition Register (AIDR)

Table 6-5 describes AIDR fields.

Table 6-5. AIDR Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error.Detection of transfer error by one of the slaves interrupt definition. 0 Detection of transfer error by one of the slaves causes regular interrupt. 1 Detection of transfer error by one of the slaves causes MCP interrupt.
27	RES	Reserved transfer type. Transaction with reserved transfer type interrupt definition. 0 Transaction with reserved transfer type causes regular interrupt. 1 Transaction with reserved transfer type causes MCP interrupt.
28	ECW	External control word transfer type. Transaction with external control word transfer type interrupt definition. 0 Transaction with external control word transfer type causes regular interrupt. 1 Transaction with external control word transfer type causes MCP interrupt.
29	AO	Address only transfer type. Transaction with address only transfer type interrupt definition. 0 Transaction with address only transfer type causes regular interrupt. 1 Transaction with address only transfer type causes MCP interrupt.
30	DTO	Data time out. Data tenure time out interrupt definition. 0 Data tenure time out causes regular interrupt. 1 Data tenure time out causes MCP interrupt.
31	ATO	Address time out. Address tenure time out interrupt definition. 0 Address tenure time out causes regular interrupt. 1 Address tenure time out causes MCP interrupt.

6.2.5 Arbiter Mask Register (AMR)

Arbiter mask register (AMR) is used to mask interrupts or reset requests. Setting a mask bit enables the corresponding interrupt or reset request; clearing a bit masks it. Regular interrupts, MCP interrupts and reset requests can be masked by AMR register. Figure 6-5 shows the fields of AMR.

Offset 0x14 Access: User read/write

	0	25	26	27	28	29	30	31
R	—		ETEA	RES	ECW	AO	DTO	ATO
W								
Reset	All zeros							

Figure 6-5. Arbiter Mask Register (AMR)

Table 6-6 describes AMR fields.

Table 6-6. AMR Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error. Detection of transfer error by one of the slaves interrupt mask bit. 0 Detection of transfer error by one of the slaves interrupt disabled. 1 Detection of transfer error by one of the slaves interrupt enabled.
27	RES	Reserved transfer type.Transaction with reserved transfer type interrupt mask bit. 0 Transaction with reserved transfer type interrupt disabled. 1 Transaction with reserved transfer type interrupt enabled.
28	ECW	External control word transfer type.Transaction with external control word transfer type interrupt mask bit. 0 Transaction with external control word transfer type interrupt disabled. 1 Transaction with external control word transfer type interrupt enabled.
29	AO	Address only transfer type. Transaction with address only transfer type interrupt mask bit. 0 Transaction with address only transfer type interrupt disabled. 1 Transaction with address only transfer type interrupt enabled.
30	DTO	Data time out. Data tenure time out interrupt mask bit. 0 Data tenure time out interrupt disabled. 1 Data tenure time out interrupt enabled.
31	ATO	Address time out. Address tenure time out interrupt mask bit. 0 Address tenure time out interrupt disabled. 1 Address tenure time out interrupt enabled.

6.2.6 Arbiter Event Attributes Register (AEATR)

Arbiter event attributes register (AEATR) reports the type of transaction that causes error, which is specified in the event register. See Section 6.2.3, “Arbiter Event Register (AER),” for more information. AEATR is cleared only by power-on reset. The attributes of the first error event are stored. Note that this means that AEATR does not change its value when AER is not clear. As AEATR is not affected by soft or hard reset, software can read this register and determine the cause of the bus failure, even if the failure caused a deadlock situation. Refer to Section 6.4.2, “Error Handling Sequence” for more information.

Figure 6-6 shows the fields of AEATR.

Offset 0x18 Access: User read/write

	0	4	5 7	8 10	11 15	16 19	20	21 23	24 26	27 31
R W	—		EVENT	—	MSTR_ID	—	TBST	TSIZE	—	TTYPE
Reset	All zeros									

Figure 6-6. Arbiter Event Attributes Register (AEATR)

Table 6-7 describes AEATR fields.

Table 6-7. AEATR Field Descriptions

Bits	Name	Description
0–4	—	Write reserved, read = 0
5–7	EVENT	Event type. 000 Address time out 001 Data time out 010 Address only transfer type 011 External control word transfer type 100 Reserved transfer type 101 Transfer error 11*c* Reserved
8–10	—	Write reserved, read = 0
11–15	MSTR_ID	Master Id. 00000 e300 core data transaction 00001 Reserved 00010 e300 core instruction fetch 00011–00111 Reserved 01000 Encryption core 01001 I^2C (boot sequencer) 01010 JTAG 01011 TPR 01100 Reserved 01101 PCI 01110 Reserved 01111 DMA 10000–11111 QUICC Engine block as follows: 100xy are used by the QUICC Engine block as follows: x = MSTR_ID[3] Least significant bit of SNUM (See Chapter 19, “Configuration,” for a description of SNUM). y = MSTR_ID[4] CETM bit from RBMR/TBMR registers. RBMR/TBMR registers are described in protocol chapters. 101xx Reserved 11xxxReserved **Note:** Master Id reflects the source of transaction and is used for debug purposes.
16–19	—	Write reserved, read = 0
20	TBST	Transfer burst. 0 Burst transaction. Transfer size is greater than 8 bytes 1 Single-beat transaction. Transfer size is up to 8 bytes

Table 6-7. AEATR Field Descriptions (continued)

Bits	Name	Description
21–23	TSIZE	Transfer size. Transfer size encoding depends on the value of the field TBST. TBST = 1: 001 1 byte 010 2 bytes 011 3 bytes 100 4 bytes 101 5 bytes 110 6 bytes 111 7 bytes 000 8 bytes TBST = 0: 000 16 bytes 001 24 bytes 010 32 bytes 011–111 Reserved
24–26	—	Write reserved, read = 0
27–31	TTYPE	Transfer type. 00000 Address-only 00001 Address-only 00010 Single-beat or burst write 00011 Reserved 00100 Address-only 00101 Reserved 00110 Burst write 00111 Reserved 0100x Address-only 0101x Single-beat or burst read 0110x Address-only 01110 Burst read 01111 Reserved 10000 Address-only 1XX01 Reserved 10010 Single-beat write 1XX11 Reserved 10100 **ecowx**—Illegal single-beat write 10110 Reserved 11000 Address-only 11010 Single-beat or burst read 11100 **eciwx**—Illegal single-beat read 11110 Burst read

6.2.7 Arbiter Event Address Register (AEADR)

Arbiter event address register (AEADR) reports the address of transaction that causes the error, which is specified in the event register. See Section 6.2.3, “Arbiter Event Register (AER),” for more information. AEADR is cleared only by power-on reset. The address of the first error event is stored. Note that this means that AEADR does not change its value when AER is not clear. As AEADR is not effected by soft or hard reset, software can read this register and determine the cause of the bus failure, even if the bus failure had caused a deadlock situation. Refer to Section 6.4.2, “Error Handling Sequence,” for more information.

Figure 6-7 shows the fields of AEADR.

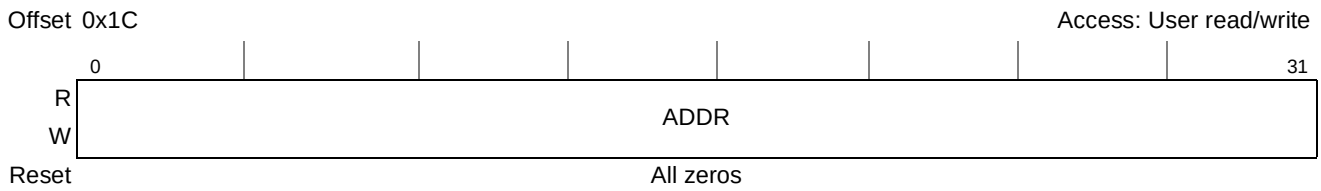


Figure 6-7. Arbiter Event Address Register (AEADR)

Table 6-8 describes AEADR fields.

Table 6-8. AEADR Field Descriptions

Bits	Name	Description
0–31	ADDR	Address of the event reported in AEATR register. See Section 6.2.6, "Arbiter Event Attributes Register (AEATR)," for more information.

6.2.8 Arbiter Event Response Register (AERR)

Arbiter event response register (AERR) determines whether different error conditions cause interrupt or reset request. Setting a bit defines the corresponding error condition to cause reset request; clearing a bit defines the corresponding error condition to cause interrupt. Figure 6-8 shows the fields of AERR.

Offset 0x20 Access: User read/write

	0	25	26	27	28	29	30	31
R	—		ETEA	RES	ECW	AO	DTO	ATO
W								
Reset	All zeros							

Figure 6-8. Arbiter Event Response Register (AERR)

Table 6-9 describes AERR field.

Table 6-9. AERR Field Descriptions

Bits	Name	Description
0–25	—	Write reserved, read = 0
26	ETEA	Transfer error. Detection of transfer error by one of the slaves event response. 0 Detection of transfer error by one of the slaves causes interrupt. 1 Detection of transfer error by one of the slaves causes reset request.
27	RES	Reserved transfer type. Transaction with reserved transfer type interrupt definition. 0 Transaction with reserved transfer type causes interrupt. 1 Transaction with reserved transfer type causes reset request.
28	ECW	External control word transfer type. Transaction with external control word transfer type interrupt definition. 0 Transaction with external control word transfer type causes interrupt. 1 Transaction with external control word transfer type causes reset request.
29	AO	Address only transfer type. Transaction with address only transfer type interrupt definition. 0 Transaction with address only transfer type causes interrupt. 1 Transaction with address only transfer type causes reset request.
30	DTO	Data time out. Data tenure time out interrupt definition. 0 Data tenure time out causes interrupt. 1 Data tenure time out causes reset request.
31	ATO	Address time out. Address tenure time out interrupt definition. 0 Address tenure time out causes interrupt. 1 Address tenure time out causes reset request.

6.3 Functional Description

The following sections describe arbiter functionality: arbitration policy and bus error detection.

6.3.1 Arbitration Policy

The arbitration process involves the masters and the arbiter. Masters arbitrate on the privilege to own an address tenure. For data tenures, the arbiter uses the same order of transactions as address tenures. Figure 6-9 shows the interface signals between the arbiter and masters that are involved in address bus arbitration.

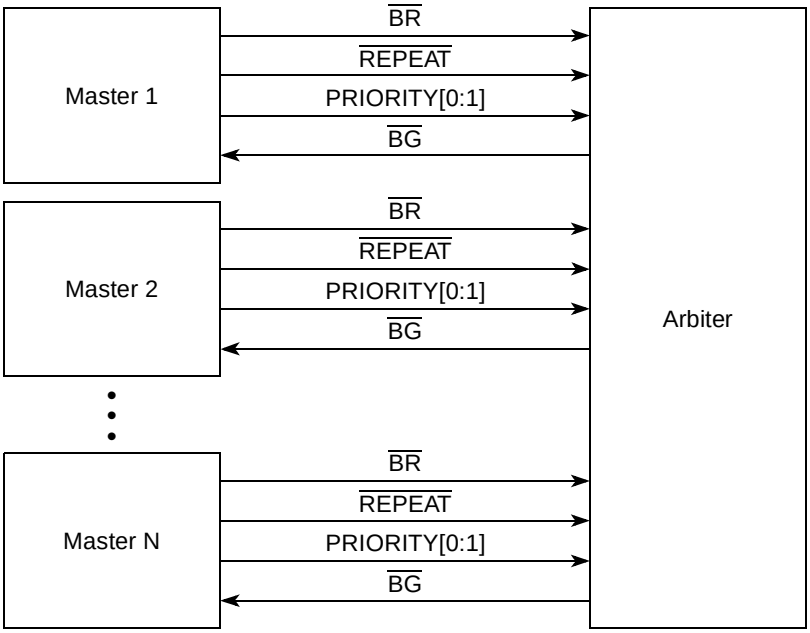


Figure 6-9. Address Bus Arbitration

A master has to acquire address bus ownership before it starts any transaction. The master asserts its own bus request signal along with the arbitration attribute signals REPEAT & PRIORITY[0:1]. The arbiter later asserts the corresponding address bus grant signal to the requesting master depending on the system states and arbitration scheme. See Section 6.3.1.1, “Address Bus Arbitration with PRIORITY[0:1],” for details on arbitration scheme. When address bus grant is received the master can start the address tenure.

6.3.1.1 Address Bus Arbitration with PRIORITY[0:1]

Whenever a master asserts its bus request to acquire address bus ownership, it can drive its PRIORITY[0:1] signals to indicate request priority. The master would be served sooner because of its higher priority level. The arbiter takes this extra information into consideration in order to yield better service for a higher priority request than a lower priority request. Therefore, the arbiter operates according to the following priority based arbitration scheme:

1. For every priority level a fair arbitration scheme is used (a simple round robin scheme)

2. For every priority level other than 0, one place is reserved as a place holder for lower level arbitration rings.
3. Each master can change its priority level at any time.

Figure 6-10 shows an example of priority-based arbitration algorithm with four priority levels. In this example, if all masters request the bus continuously, the following order of bus grants occurs with the specific bandwidth:

- M6 gets 1/2 of the bus bandwidth
- M4 and M5 each gets 1/6 of the bus bandwidth
- M0 and M3 each gets 1/18 of the bus bandwidth
- M1 and M2 each gets 1/36 of the bus bandwidth

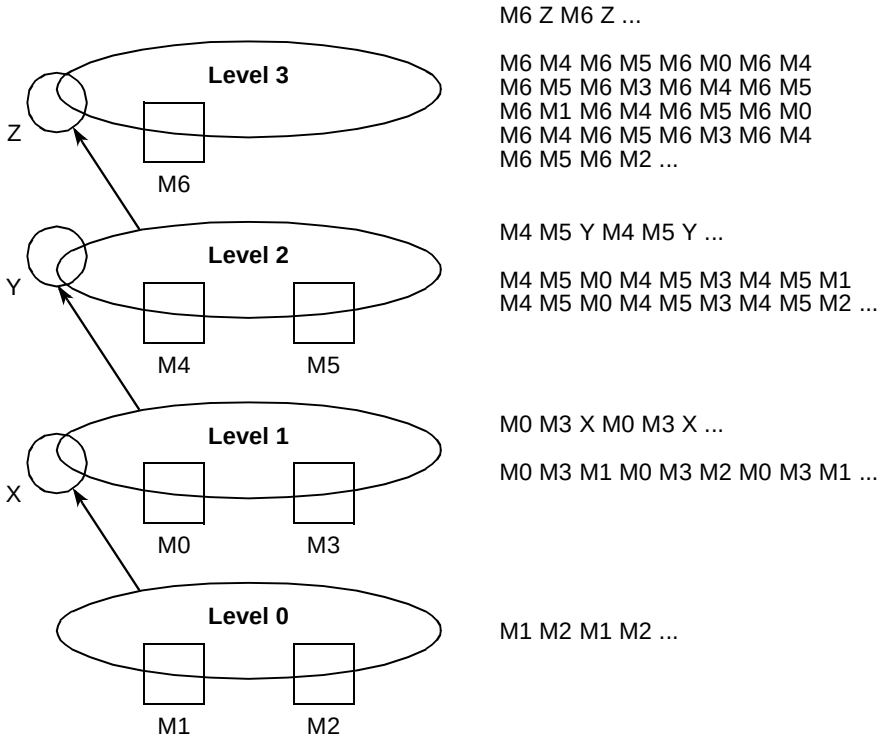


Figure 6-10. An Example of Priority-Based Arbitration Algorithm

NOTE

See each bus master's chapter and Section 5.3.2.4, "System Priority and Configuration Register (SPCR)," for more details about priority programming.

6.3.1.2 Address Bus Arbitration with $\overline{\text{REPEAT}}$

When a master owns the address bus and wants to perform another transaction, it can assert bus request along with $\overline{\text{REPEAT}}$, to make a repeat request to the arbiter. Consequently, the arbiter asserts bus grant to the same master if the current address tenure is not being $\overline{\text{ARTRY}}$ed. This happens regardless of the priority level of bus request from other masters. In another word, "repeat request" overrides the priority scheme.

Even though repeat request can improve the page hit ratio and the overall memory bandwidth efficiency, it can increase the worst case latency of individual master. Therefore, the arbiter has programmable counter to limit the maximum number of consecutive transactions that are performed by masters. Whenever the counter expires, arbiter ignores the $\overline{\text{REPEAT}}$ signal and falls back to the regular arbitration scheme. PCI master has a dedicated repeat counter as it might need more repeated transactions before accepting read requests. PCI ordering rules require that the PCI bridge should empty all queued write operations before any new read operation can begin. See Section 3.2.5, “Transaction Ordering and Posting,” of the *PCI Local Bus Specifications Rev 2.2* for more information.

See Section 6.2.1, “Arbiter Configuration Register (ACR),” for more details about programming ACR[RPTCNT] and ACR[PCI_RPTCNT].

6.3.1.3 Address Bus Arbitration after $\overline{\text{ARTRY}}$

The $\overline{\text{ARTRY}}$ protocol is used primarily by the CPU to interrupt a transaction that hits to a modified line in its D-cache, so that it can maintain data coherency by performing the snoop copyback. When CPU asserts $\overline{\text{ARTRY}}$, the bus is immediately granted to the CPU to perform snoop copyback. After the completion of snoop copyback, the arbiter grants the bus back to the master that had its transaction $\overline{\text{ARTRY}}$ed.

6.3.1.4 Address Bus Parking

The arbiter supports address bus parking. This feature implies that when no master is requesting the bus (all bus requests are negated), the arbiter can choose to park the address bus (or assert the address bus grant) to a master. The parked master can skip the bus request and assume the bus mastership directly. This reduces the access latency for parked master.

See Section 6.2.1, “Arbiter Configuration Register (ACR),” for more details about ACR[APARK] and ACR[PARKM].

6.3.1.5 Data Bus Arbitration

For every committed address tenure a data tenure is required to complete the transaction.

In the device system, the arbiter controls the issuing of data bus grants to a master and a slave, which are involved in a data tenure of a previously performed address tenure.

6.3.2 Bus Error Detection

The arbiter is responsible for tracking the following cases on the bus:

- Address time out
- Data time out
- Transfer error
- Address only transaction type
- Reserved transaction type
- Illegal (**eciwx**/**ecowx**) transaction type

6.3.2.1 Address Time Out

Address time out occurs, if the address tenure was not ended before the specified time-out period (programmed by ATR[ATO]). In this case, the arbiter performs as follows:

1. Ends the address tenure.
2. Starts data tenure and ends it by asserting transfer error.
3. Reports on the event to AER[ATO].
4. Issues reset request, MCP or regular interrupt according to AERR[ATO] and AIDR[ATO], if enabled by AMR[ATO].
5. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.2 Data Time Out

Data time out occurs, if the data tenure was not ended before the specified time-out period (programmed by ATR[DTO]). In this case, the arbiter performs as follows:

1. Ends the data tenure by asserting transfer error.
2. Reports on this event in AER[DTO].
3. Issues reset request, MCP or regular interrupt according to AERR[DTO] and AIDR[DTO], if enabled by AMR[DTO].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.3 Transfer Error

The arbiter tracks the transfer error asserted by one of the slaves. In this case, the arbiter performs as follows:

1. Reports on the event to AER[ETEA].
2. Issues reset request, MCP or regular interrupt according to AERR[ETEA] and AIDR[ETEA] if enabled by AMR[ETEA].
3. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.4 Address Only Transaction Type

Table 6-10 shows transaction types, which are defined as address only:

Table 6-10. Address Only Transaction Type Encoding

ttype[0:4]	Bus Commands
00000	Clean block
00100	Flush block
01000	Sync
01100	Kill block
10000	eieio
11000	TLB Invalidate

Table 6-10. Address Only Transaction Type Encoding (continued)

ttype[0:4]	Bus Commands
00001	lwarx reservation set
01001	tlbsync
01101	icbi

The arbiter allows address-only (AO) transactions on the bus and the G2 core has ability to issue address-only (AO) transactions (see HID0 [ABE] in the *G2 PowerPC Core Reference Manual*, Rev 1). As there is no advantage in using AO transaction in this system, the bus monitor allows the detection of AO transactions and treats them as an error.

The transaction with an address only transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[AO].
3. Issues reset request, MCP or regular interrupt according to AERR[AO] and AIDR[AO] if enabled by AMR[AO].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.5 Reserved Transaction Type

Table 6-11 shows transaction types defined as reserved:

Table 6-11. Reserved Transaction Type Encoding

ttype[0:4]	Bus Commands
00101	Reserved
1xx01	Reserved for customer
10110	Reserved
00011	Reserved
00111	Reserved
01111	Reserved
1xx11	Reserved for customer

The transaction with a reserved transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[RES].
3. Issues reset request, MCP or regular interrupt according to AERR[RES] and AIDR[RES], if enabled by AMR[RES].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.6 Illegal (eciwx/ecowx) Transaction Type

Table 6-12 shows transaction types defined as illegal.

Table 6-12. Illegal Transaction Type Encoding

ttype[0:4]	Bus command
10100	External control word write (**ecowx**)
11100	External control word read (**eciwx**)

The transaction with an illegal (eciwx, ecowx) transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Starts data tenure and ends data tenure by asserting $\overline{\text{TEA}}$.
3. Reports on the event in AER[ECW].
4. Issues reset request, MCP or regular interrupt according to AERR[ECW] and AIDR[ECW], if enabled by AMR[ECW].
5. Updates transaction attributes and address of AEATR and AEADR for the first error event.

See Section 6.2.3, “Arbiter Event Register (AER),” Section 6.2.4, “Arbiter Interrupt Definition Register (AIDR),” Section 6.2.5, “Arbiter Mask Register (AMR),” Section 6.2.6, “Arbiter Event Attributes Register (AEATR),” Section 6.2.7, “Arbiter Event Address Register (AEADR),” and Section 6.2.8, “Arbiter Event Response Register (AERR),” for more information.

6.4 Initialization/Applications Information

The following sections describe the initialization and error handling sequences for the arbiter.

6.4.1 Initialization Sequence

The following initialization sequence is recommended:

1. Write to ACR to configure pipeline depth, address bus parking mode, global maximum repeat count, PCI maximum repeat count.
2. Write to AERR defines whether different error events cause a reset request or an interrupt.
3. Write to AIDR defines the kind of interrupt (regular or MCP) caused by each error event. Note that this is necessary only if interrupts are enabled and AERR defines error events to cause interrupt.
4. Write to AMR to enable interrupts.
5. Write to ATR to set the ATO and DTO timers. Note that this is only necessary if the required timers are less than the maximum value (which is default).

6.4.2 Error Handling Sequence

The following error handling sequence is recommended:

1. Read to AER to find out about the error that occurred in the system. Also, read the values of AEATR and AEADR to check on the first error event in the system.

2. If those registers are not accessible because of a bus deadlock situation, reset the chip and read the values of the AEATR and AEADR registers to check on the event that causes this problem to the system. Use $\overline{\text{HRESET}}$ to reset the chip to guarantee that the information stored in AEATR and AEADR is not lost.
3. Clear all the previous events by writing ones to the AER. This register is also cleared after reset.

Chapter 7
e300 Processor Core Overview

This chapter provides an overview of features for the embedded microprocessors in the e300 core family, which are PowerPC microprocessors built on Power Architecture technology. Throughout this chapter, the terms ‘e300 core’, ‘core’, and ‘processor’ are used interchangeably. The term‘e300c2’ is used when describing an implementation-specific feature or when a difference exists between different configurations. The term ‘e300’ is used when describing a feature that pertains to the family of e300 processors. The MPC8323E uses an e300c2 core.

7.1 Overview

This section describes the details of the e300 core, provides a block diagram showing the major functional units, and briefly describes how these units interact. All differences between the e300 and previous PowerPC implementations derived from the MPC603e processor are noted. For additional information, please refer to the *e300 PowerPC Core Family Reference Manual*.

The e300 core is a low-power implementation of this microprocessor family of reduced instruction set computing (RISC) microprocessors. The core implements the 32-bit portion of the PowerPC architecture, which defines 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The core is a superscalar processor that can issue and retire as many as three instructions per clock cycle. Instructions can execute out of program order for increased performance; however, the core makes completion appear sequential.

The e300 core integrates independent execution units including: an integer unit (IU), a branch processing unit (BPU), a load/store unit (LSU), and a system register unit (SRU). The e300c2integrates an additional integer unit for a total of two IUs. Note that the e300c2 does not include an FPU. The ability to execute instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for e300-core-based systems. Most integer instructions execute in one clock cycle. The additional IUs along with enhanced multipliers in the e300c2improve multiply instructions to a maximum two-cycle latency, a significant improvement from previous processors.

Figure 7-1 is a block diagram of the e300c2 core. Note that it does not support floating-point operations.

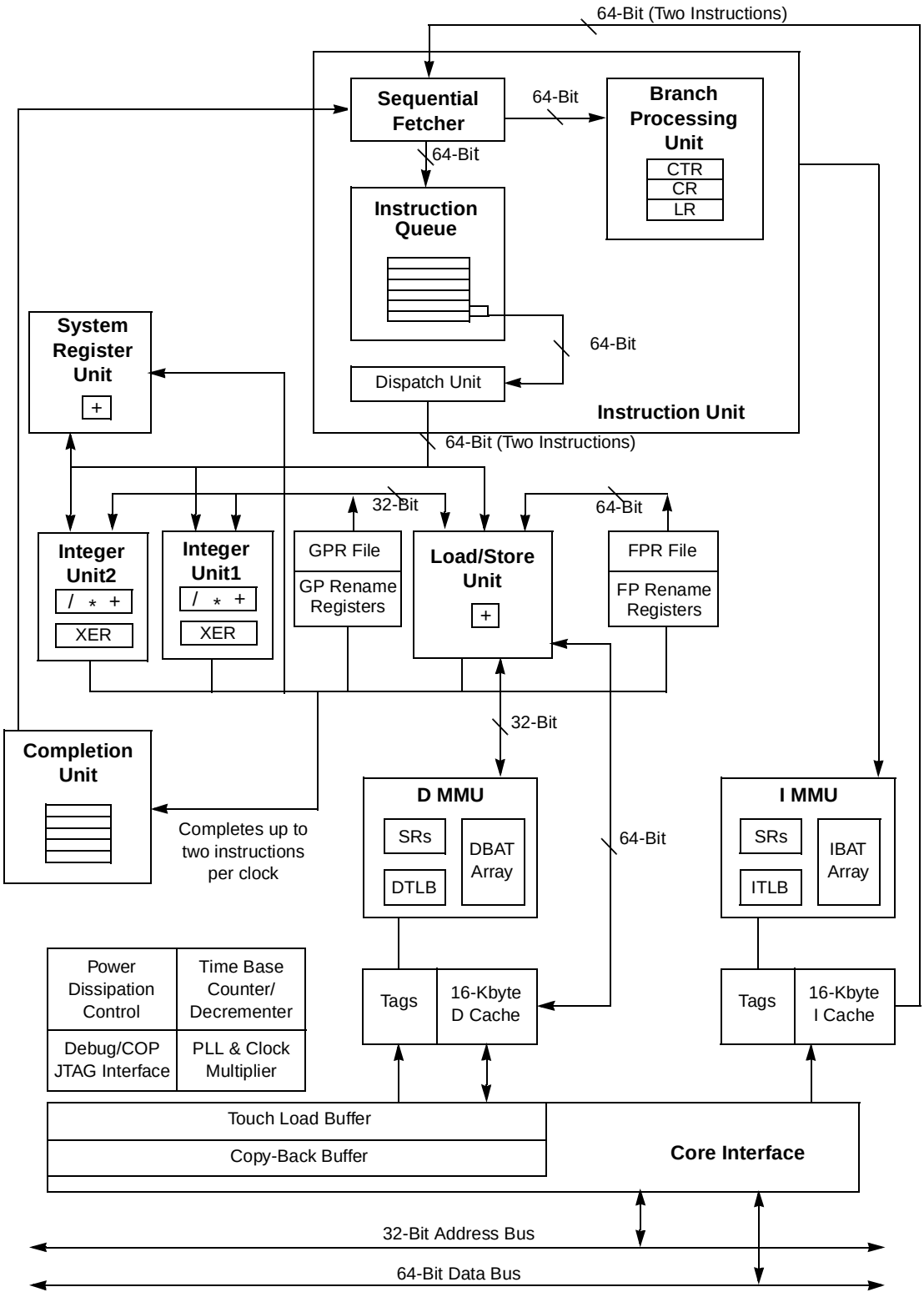


Figure 7-1. e300c2 Core Block Diagram

The e300c2 includes 16-Kbyte, four way set-associative instruction and data caches. The MMUs contain 64-entry, two-way, set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged, virtual-memory, address translation, and variable-sized block translation. The TLBs use a least recently used (LRU) replacement algorithm and the caches use a pseudo least recently used algorithm (PLRU).

The core also supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays, each containing eight pairs of BATs, an increase from four pairs of each type of BATs in the G2 core. This increase provides more flexibility in protecting accesses and providing translation on a segment, block, or page basis for memory accesses and I/O accesses. Effective addresses are compared simultaneously with all eight entries in the BAT array during block translation. In accordance with the PowerPC architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

As part of the coherent system bus (CSB), the e300 core has a 64-bit data bus and a 32-bit address bus. During normal operation, the e300 core provides a modified, exclusive, and invalid coherency protocol which is a compatible subset of a four-state (modified/exclusive/shared/invalid) MESI protocol. However, the e300 data cache contains a programmable MESI extension that supports the shared cache coherency state (similar to other PowerPC processors). Both protocols operate coherently in systems that contain four-state caches. Although MESI is supported by the e300 core, it is not implemented on the MPC8323E. The core also supports single-beat and burst data transfers for memory accesses and supports memory-mapped I/O operations.

The true little-endian mode is another enhanced capability of the e300 core. Unlike the PowerPC little-endian mode (which manipulates only the address bits), no longer supported on the e300, the true little-endian mode actually operates on true little-endian instructions and data from memory.

The critical interrupt is an additional interrupt in the e300 core and has higher priority order than the system management interrupt. Also, debug features are improved in the e300. Additional SPRG interrupt handling registers are provided for enhancing flexibility for the operating system.

7.1.1 Features

This section describes the major features of the e300 core:

- High-performance, superscalar microprocessor core
  - As many as three instructions issued and retired per clock (two instructions plus one branch instruction)
  - As many as five instructions in execution per clock
  - Single-cycle execution for most instructions
- Independent execution units and two register files
  - Branch processing unit (BPU) featuring static branch prediction
  - Two 32-bit integer units (IU) in the e300c2.
  - Load/store unit (LSU) for data transfer between data-cache and general-purpose registers (GPRs) and floating-point registers (FPRs)

  - System register unit (SRU) that executes condition register (CR), special-purpose register (SPR), and integer add/compare instructions. Add/compare instructions are also executed in the IUs.
  - Thirty-two 32-bit GPRs for integer operands
  - Thirty-two 64-bit FPRs for single- or double-precision operands
- High instruction and data throughput
  - Zero-cycle branch capability (branch folding)
  - Programmable static branch prediction on unresolved conditional branches
  - Two integer units with enhanced multipliers in thee300c2 for increased integer instruction throughput and a maximum two-cycle latency for multiply instructions
  - Instruction fetch unit capable of fetching two instructions per clock from the instruction cache
  - A six-entry instruction queue (IQ) that provides lookahead capability
  - Independent pipelines with feed-forwarding that reduces data dependencies in hardware
  - 16-Kbyte, four-way set-associative instruction and data caches on the e300c2.
  - Cache write-back or write-through operation programmable on a per-page or per-block basis
  - Features for instruction and data cache locking and protection
  - BPU that performs CR lookahead operations
  - Address translation facilities for 4-Kbyte page size, variable block size, and 256-Mbyte segment size
  - A 64-entry, two-way, set-associative ITLB and DTLB
  - Eight-entry data and instruction BAT arrays providing 128-Kbyte to 256-Mbyte blocks
  - Software table search operations and updates supported through fast trap mechanism
  - 52-bit virtual address; 32-bit physical address
- Facilities for enhanced system performance
  - A 64-bit split-transaction internal data bus interface to the coherent system bus (CSB) with burst transfers
  - Support for one-level address pipelining on the CSB interface
  - True little-endian mode for compatibility with other true little-endian devices
  - Critical interrupt support
  - Hardware support for misaligned little-endian accesses
  - Configurable processor bus frequency multipliers as defined in the *MPC8323E Integrated Processor Hardware Specifications*
- Integrated power management
  - Internal processor/bus clock multiplier ratios
  - Three power-saving modes: doze, nap, and sleep
  - Automatic dynamic power reduction when internal functional units are idle
- In-system testability and debugging features through JTAG boundary-scan capability

Features specific to the e300 core not present on the G2 processors follow:

- Enhancements to the register set
  - The e300 core has one more HID0 bit than the G2:
    - The enable cache parity checking (ECPE) bit, HID0[1], gives the e300 core the ability to enable the taking of a machine check interrupt based on the detection of a cache parity error
- Enhancements to cache implementation
  - 16-Kbyte, four-way set-associative instruction and data caches on the e300c2.
  - Full parity checking is performed on both instruction and data cache memory arrays
  - Lockable L1 instruction and data caches—entire cache or on a per-way basis up to 3 of 4 ways on the e300c2
  - New **icbt** instruction supports initialization of instruction cache
  - Data cache supports four-state MESI coherency protocol (not implemented on MPC8323E)
  - The instruction cache is blocked only until the critical load completes (hit under reloads allowed)
  - Instruction cancel mechanism improves utilization of instruction cache by supporting hits-under-cancels and misses-under-cancels.
  - The critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.
  - Data cache queue sharing makes cast-outs and snoop pushes more efficient
  - Provides for an optional data cache operation broadcast feature (enabled by HID0[ABE]) that allows for coherent system management. All of the data cache control instructions, except **dcbz** (**dcbi**, **dcbf**, and **dcbst**) require that HID0[ABE] be enabled to broadcast.
  - Instruction fetch burst feature allows all instruction fetches from caching-inhibited space to be performed on the bus as burst transactions
- Interrupts
  - The e300 core offers hardware support for misaligned little-endian accesses. Little-endian load/store accesses that are not on a word boundary, except for strings and multiples, generate interrupts under the same circumstances as big-endian accesses.
  - The e300 core supports true little-endian mode to minimize the impact on software porting from true little-endian systems.
  - An input interrupt signal, $\overline{cint}$, is provided to trigger the critical interrupt exception on the e300 core.
- Bus clock—PLL configuration signals include seven signals for settings and control: *pll_cfg[0:6]*.
- Debug features
  - Breakpoint status recorded in DBCR and IBCR control registers
  - Two signals for the debug interface: $\overline{stopped}$ and $\overline{ext\_halt}$

Figure 7-1 provides a block diagram of the e300 core that shows how the execution units—IU, BPU, LSU, and SRU—operate independently and in parallel. It should be noted that this is a conceptual diagram and does not attempt to show how these features are physically implemented on the device.

The e300 core provides address translation and protection facilities, including an ITLB, DTLB, and instruction and data BAT arrays. Instruction fetching and issuing are handled in the instruction unit.

Translation of addresses for cache or external memory accesses are handled by the MMUs. Both units are discussed in more detail in Section 7.1.2, “Instruction Unit,” and Section 7.1.5.1, “Memory Management Units (MMUs).”

7.1.2 Instruction Unit

As shown in Figure 7-1 the e300 core instruction unit, containing a fetch unit, instruction queue, dispatch unit, and BPU, provides centralized control of instruction flow to the execution units. The instruction unit determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

The instruction unit fetches the instructions from the instruction cache into the instruction queue. The BPU receives branch instructions from the fetcher and uses static branch prediction to allow fetching from a predicted instruction stream while a conditional branch is evaluated. The BPU folds out for unconditional branch instructions and conditional branch instructions unaffected by instructions in the execution pipeline.

Instructions issued beyond a predicted branch cannot complete execution until the branch is resolved, preserving the programming model of sequential execution. If any of these are branch instructions, they are decoded but not issued. Instructions to be executed by the FPU, IU, LSU, and SRU are issued and allowed to progress up to the register write-back stage. Write-back is allowed when a correctly predicted branch is resolved, and execution continues along the predicted path.

If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

7.1.2.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 7-1, holds as many as six instructions and loads up to two instructions from the instruction unit during a single cycle. The instruction fetch unit continuously loads as many instructions as space in the IQ allows. Instructions are dispatched to their respective execution units from the dispatch unit at a maximum rate of two instructions per cycle. Dispatching is facilitated to the IUs, FPU, LSU, and SRU by the provision of a reservation station at each unit. The dispatch unit performs source and destination register dependency checking, determines dispatch serializations, and inhibits subsequent instruction dispatching as required.

For a more detailed overview of instruction dispatch, see Section 7.3.6, “Instruction Timing.”

7.1.2.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the fetch unit and performs CR lookahead operations on conditional branches to resolve them early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the core fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an adder to compute branch target addresses and three user-control registers: the link register (LR), the count register (CTR), and the conditional register (CR). The BPU calculates the return

pointer for sub-routine calls and saves it into the LR for certain types of branch instructions. The LR also contains the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction. The CTR contains the branch target address for the Branch Conditional to Count Register (**bcctr***x*) instruction. The contents of the LR and CTR can be copied to or from any GPR. Because the BPU uses dedicated registers rather than GPRs or FPRs, execution of branch instructions is largely independent from execution of integer and floating-point instructions.

7.1.3 Independent Execution Units

The PowerPC architecture's support for independent execution units allows implementation of processors with out-of-order instruction execution. For example, because branch instructions do not depend on GPRs or FPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

The four other execution units and the completion unit are described in the following sections.

7.1.3.1 Integer Unit (IU)

The IU executes all integer instructions. The IU executes one integer instruction at a time, performing computations with its arithmetic logic unit (ALU), multiplier, divider, and XER register. Most integer instructions are single-cycle instructions. The 32 GPRs hold integer operands. Stalls due to contention for GPRs are minimized by the automatic allocation of rename registers. The core writes the contents of the rename registers to the appropriate GPR when integer instructions are retired by the completion unit. The e300c2 provides two integer units for greater integer instruction throughput along with enhanced multipliers in each IU for faster multiply-instruction execution.

7.1.3.2 Load/Store Unit (LSU)

The LSU executes all load and store instructions and provides the data transfer interface between the GPRs, FPRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and provides sequencing for load/store string and multiple instructions.

Load and store instructions are issued and executed in program order; however, the memory accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering.

Cacheable loads, when free of data bus dependencies, can execute out of order with a maximum throughput of one per cycle and with a two-cycle total latency. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR or FPR. Stores cannot be executed in a predicted manner and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The core executes store instructions with a maximum throughput of one per cycle and with a three-cycle total latency. The time required to perform the actual load or store depends on whether the operation involves the cache, system memory, or an I/O device.

7.1.3.3 System Register Unit (SRU)

The SRU executes various system-level instructions, including condition register logical operations and move to/from special-purpose register instructions. It also executes integer add/compare instructions. In order to maintain system state, most instructions executed by the SRU are completion-serialized; that is,

the instruction is held for execution in the SRU until all prior instructions issued have completed. Results from completion-serialized instructions executed by the SRU are not available or forwarded for subsequent instructions until they complete.

7.1.4 Completion Unit

The completion unit tracks instructions in program order from dispatch through execution and then completes. Completing an instruction commits the core to any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the core must recover from a mispredicted branch or an interrupt.

Instruction state and other information required for completion is kept in a five-entry FIFO completion queue. A single completion queue entry is allocated for each instruction once it enters the execution unit from the dispatch unit. An available completion queue entry is a required resource for dispatch; if no completion entry is available, dispatch stalls. A maximum of two instructions per cycle are completed in order from the queue.

7.1.5 Memory Subsystem Support

The core provides separate instruction and data caches and MMUs. The core also provides an efficient processor bus interface to facilitate access to main memory and other bus subsystems. The memory subsystem support functions are described in the following sections.

7.1.5.1 Memory Management Units (MMUs)

The core MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gigabytes (2^{32}) of physical memory (referred to as real memory in the architecture specification) for instruction and data. The MMUs also control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the processor for each page to assist implementation of a demand-paged virtual memory system. Note that software assistance is required for the device to maintain reference and changed status. A key bit is implemented to provide information about memory protection violations prior to page table search operations.

The LSU calculates effective addresses for data loads and stores, performs data alignment to and from cache memory, and provides the sequencing for load and store string and multiple word instructions. The instruction unit calculates effective addresses for instruction fetching.

After an EA is generated, its higher-order bits are translated by the appropriate MMU into physical address bits. The lower-order EA bits are the same on the physical address which are directed to the on-chip cache and formed the index into a four-way set-associative tag array. After translating the address, the MMU passes the higher-order physical address bits to the cache and the cache lookup completes. For caching-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32-bit physical address is then used by the memory unit and the core interface to access external memory.

The MMU also directs the address translation and enforces the protection hierarchy programmed by the operating system in relation to the supervisor/user privilege level of the access and in relation to whether the access is a load or store.

For instruction fetches, the IMMU looks for the address in the ITLB and in the IBAT array. If an address hits both, the IBAT array translation is used. Data accesses cause a lookup in the DTLB and DBAT array. In most cases, the translation is in a TLB and the physical address bits are available to the on-chip cache.

The e300 core implements four more IBAT and four more DBAT entries than the G2.

When the EA misses in the TLBs, the core provides hardware assistance for software to perform a search of the translation tables in memory. The hardware assist consists of the following features:

- Automatic storage of the missed effective address in IMISS and DMISS
- Automatic generation of the primary and secondary hashed real addresses of the page-table entry group (PTEG), which are readable from the HASH1 and HASH2 register locations.

  The HASH data is generated from the contents of the IMISS or DMISS register. The register that is selected depends on the miss (instruction or data) that was last acknowledged.
- Automatic generation of the first word of the page table entry (PTE) of the tables being searched
- A real page address (RPA) register that matches the format of the lower word of the PTE
- TLB access instructions (**tlbli** and **tlbld**) that are used to load an address translation into the instruction or data TLBs
- Shadow registers for GPR0–GPR3 that allow miss code to execute without corrupting the state of any of the existing GPRs. Shadow registers are used only for servicing a TLB miss.

See Section 7.3.5.2, “Implementation-Specific Memory Management,” for more information about memory management for the core.

7.1.5.2 Cache Units

The e300c2 provides 16-Kbyte, four-way set-associative instruction and data caches. The cache block is 32 bytes long. The caches adhere to a write-back policy, but the e300 core allows control of cacheability, write policy, and memory coherency at the page and block levels. The caches use a pseudo LRU replacement policy.

As shown in Figure 7-1, the caches provide a 64-bit interface to the instruction fetch unit and LSU. The surrounding logic selects, organizes, and forwards the requested information to the requesting unit. Write operations to the cache can be performed on a byte basis, and a complete read-modify-write operation to the cache can occur in each cycle.

The load/store and instruction fetch units provide the caches with the address of the data or instruction to be fetched. In the case of a cache hit, the cache returns two words to the requesting unit.

Because the data cache tags are single-ported, simultaneous load/store and snoop accesses cause resource contention. Snoop accesses have the highest priority and are given first access to the tags, unless the snoop access coincides with a tag write; in this case the snoop is retried and must rearbitrate for cache access. Loads or stores deferred due to snoop accesses are performed on the clock cycle following the snoop.

The e300 core includes a new instruction cancel extension. The instruction cancel extension improves utilization of the instruction cache during cancel operations. It allows a new instruction fetch to be issued to the cache or to the bus if a canceled instruction fetch is pending or active on the bus. This supports hit-under-cancel and miss-under-cancel instruction fetch operations.

7.1.6 Bus Interface Unit (BIU)

Because the caches are on-chip, write-back caches, the most common transactions are burst-read memory operations, burst-write memory operations, and single-beat (noncacheable or write-through) memory read and write operations. There can also be address-only operations, variants of the burst and single-beat operations, (for example, global memory operations that are snooped and atomic memory operations), and address retry activity (for example, when a snooped read access hits a modified cache block).

Memory accesses can occur in single-beat (1–8 bytes) and four-beat burst (32 bytes) data transfers on the 64-bit data bus. The address and data buses operate independently to support pipelining and split transactions during memory accesses.

The e300 bus interface unit (BIU) has been enhanced to allow a pipeline slot to become available once a previous transaction has been granted the data bus (that is, as early as when the data tenure starts rather than after the data tenure completes), thus allowing for greater bus utilization in systems that support it. This is sometimes referred to as 1 1/2-level pipelining.

Typically, memory accesses are weakly ordered, meaning that sequences of operations, including load/store string and multiple instructions, do not necessarily complete in the order they begin. This weak ordering maximizes the efficiency of the bus without sacrificing coherency of the data. The core allows read operations to precede store operations (except when a dependency exists, or in cases where a noncacheable access is performed), and provides support for a write operation to proceed a previously queued read data tenure (for example, allowing a snoop push to be enveloped by the address and data tenures of a read operation). Because the processor can dynamically optimize run-time ordering of load/store traffic, overall performance is improved.

7.1.7 System Support Functions

The e300 core implements several support functions that include power management, time base/decrementer registers for system timing tasks, a JTAG (based on IEEE 1149.1) interface, hardware debug, and a phase-locked loop (PLL) clock multiplier. These system support functions are described in the following sections.

7.1.7.1 Power Management

The e300 core provides four power modes, selectable by setting the appropriate control bits in the machine state register (MSR) and the hardware implementation register 0 (HID0). When entering into a power mode other than full-power, the core will request entry via a $\overline{qreq}$ signal and will only enter another power mode after an acknowledge ($\overline{qack}$) is received. The four power modes are as follows:

- Full-power—This is the default power state of the e300 core. The e300 core is fully powered and the internal functional units are operating at the full processor clock speed. If the dynamic power

management mode is enabled, functional units that are idle will automatically enter a low-power state without affecting performance, software execution, or external hardware.

- Doze—All the functional units of the e300 core are disabled except for the time base/decrementer registers and the bus snooping logic. When the processor is in doze mode, an external asynchronous interrupt, system management interrupt, decrementer interrupt, hard or soft reset, or machine check brings the e300 core into the full-power state. The core in doze mode maintains the PLL in a fully-powered state and locked to the system external clock input (*sysclk*), so a transition to the full-power state takes only a few processor clock cycles.
- Nap—The nap mode further reduces power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state. The core returns to the full-power state on receipt of an external asynchronous interrupt, system management interrupt, decrementer interrupt, hard or soft reset, or machine check input ($\overline{mcp}$) signal. A return to full-power state from a nap state takes only a few processor clock cycles.
- Sleep—Sleep mode reduces power consumption to a minimum by disabling all internal functional units; then external system logic may disable the PLL and *sysclk*. Returning the core to the full-power state requires the enabling of the PLL and *sysclk*, followed by the assertion of an external asynchronous interrupt, system management interrupt, hard or soft reset, or $\overline{mcp}$ signal after the time required to relock the PLL.

7.1.7.2 Time Base/Decrementer

The time base is a 64-bit register (accessed as two 32-bit registers) that is incremented once every four bus clock cycles; external control of the time base is provided through the time base/decrementer clock base enable (*tben*) signal. The decrementer is a 32-bit register that generates a decrementer interrupt after a programmable delay. The contents of the decrementer register are decremented once every four bus clock cycles, and the decrementer interrupt is generated as the count passes through zero.

7.1.7.3 JTAG Test and Debug Interface

The core provides JTAG and hardware debug functions for facilitating board testing and chip debugging. The JTAG test interface (based on IEEE 1149.1) provides a means for boundary-scan testing of the core and the attached system logic. The hardware debug function accesses the JTAG test port, providing a means for executing test routines and facilitating chip and software debugging.

All instruction and data address breakpoints are accessible in the IBCR and DBCR. See Section 7.3.8, “Debug Features,” for more information.

7.1.7.4 Clock Multiplier

The internal clocking of the e300 core is generated from and synchronized to the external clock signal, *sysclk*, by means of a voltage-controlled, oscillator-based PLL. The PLL provides programmable internal processor clock multiplier ratios which multiply the externally supplied clock frequency. The bus clock is the same frequency and is synchronous with *sysclk*. The configuration of the PLL can be read by software from the hardware implementation register 1 (HID1).

7.2 PowerPC Architecture Implementation

The PowerPC architecture consists of the following layers, and adherence to the PowerPC architecture can be measured in terms of which of the following levels of the architecture is implemented:

- User instruction set architecture (UISA)—Defines the base user-level instruction set, user-level registers, data types, floating-point interrupt model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- Virtual environment architecture (VEA)—Describes the memory model for a multiprocessor environment, defines cache control instructions, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA but may not necessarily adhere to the OEA.
- Operating environment architecture (OEA)—Defines the memory management model, supervisor-level registers, synchronization requirements, and interrupt model. Implementations that conform to the OEA also adhere to the UISA and VEA.

The PowerPC architecture allows a wide range of designs for such features as cache and core interface implementations.

7.3 Implementation-Specific Information

This section describes the PowerPC architecture in general and specific details about the implementation of the e300 core as a low-power, 32-bit member of this PowerPC core family. The main topics addressed are as follows:

- Section 7.3.1, “Register Model,” describes the registers for the operating environment architecture common among e300 cores that implement the PowerPC architecture and describes the programming model. It also describes the additional registers that are unique to the core.
- Section 7.3.2, “Instruction Set and Addressing Modes,” describes the PowerPC instruction set and addressing modes for the OEA, and defines and describes the instructions implemented in the core.
- Section 7.3.3, “Cache Implementation,” describes the cache model that is defined generally for cores that implement the PowerPC architecture by the VEA. It also provides specific details about the e300 core cache implementation.
- Section 7.3.4, “Interrupt Model,” describes the interrupt model of the OEA and the differences in the core interrupt model.
- Section 7.3.5, “Memory Management,” describes generally the conventions for memory management among these cores. This section also describes the core implementation of the 32-bit PowerPC memory management specification.
- Section 7.3.6, “Instruction Timing,” provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC architecture and the e300 core.
- Section 7.1.6, “Bus Interface Unit (BIU),” describes the signals implemented on the core.

The e300 core is a high-performance, superscalar processor core. The PowerPC architecture allows optimizing compilers to schedule instructions to maximize performance through efficient use of the PowerPC instruction set and register model. The multiple, independent execution units allow compilers to

optimize instruction throughput. Compilers that take advantage of the flexibility of the PowerPC architecture can additionally optimize system performance.

The following sections summarize the features of the core, including both those that are defined by the architecture and those that are unique to the various core implementations.

Specific features of the core are listed in Section 7.1.1, “Features.”

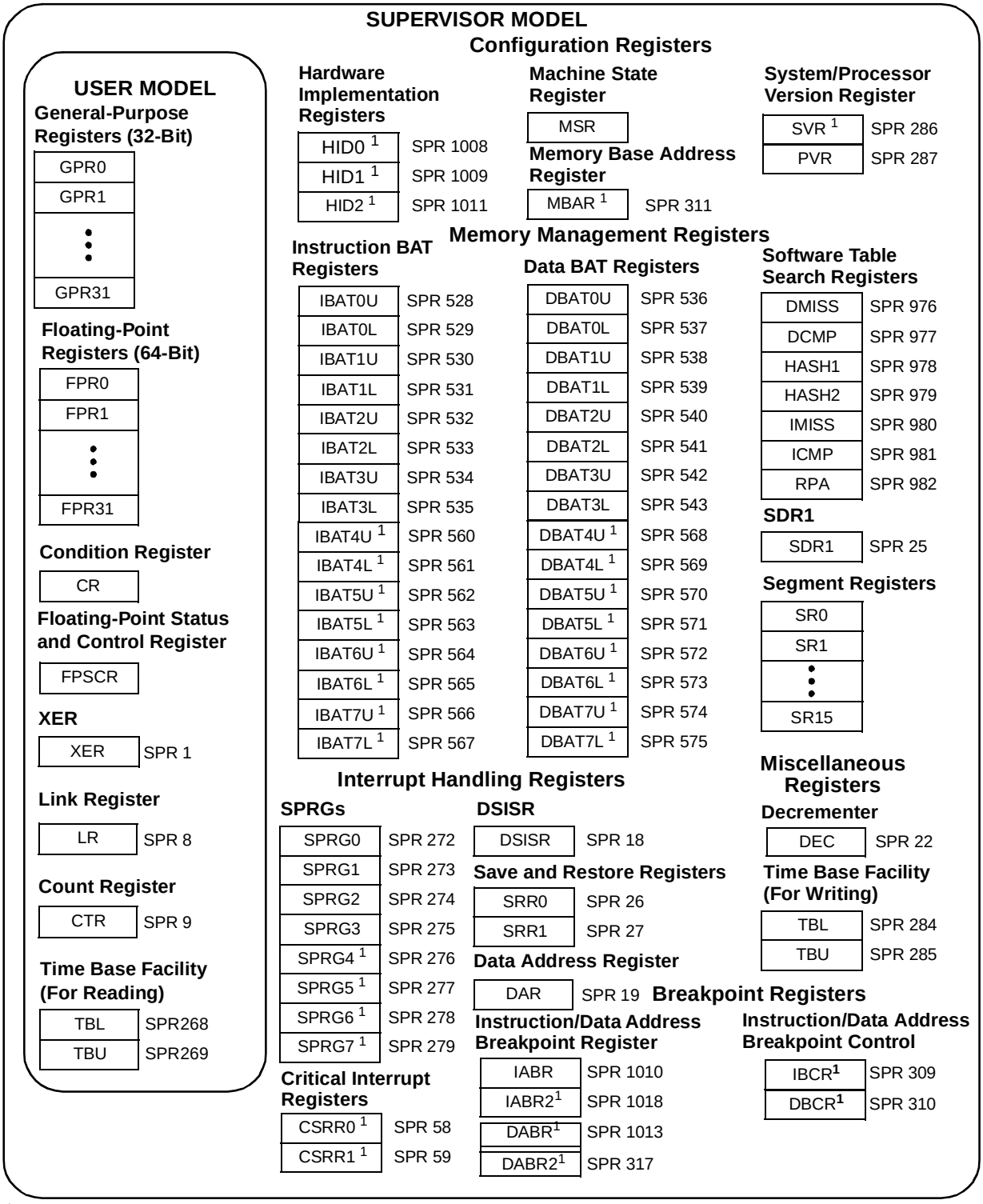
7.3.1 Register Model

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two-source operands. Load and store instructions transfer data between registers and memory.

The e300 core has two levels of privilege: supervisor mode of operation (typically used by the operating system) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers. Each core also has its own unique set of hardware implementation (HID) registers.

Having access to privileged instructions, registers, and other resources allows the operating system to control the application environment (providing virtual memory and protecting operating system and critical machine resources). Instructions that control the state of the e300 core, the address translation mechanism, and supervisor registers can be executed only when the core is operating in supervisor mode.

Figure 7-2 shows all the core registers available at the user and supervisor level. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands for the move to/from SPR instructions.



1 These registers are e300 core implementation-specific (not defined by the PowerPC architecture).

Figure 7-2. e300 Programming Model—Registers

The following sections describe the e300-core-implementation-specific features as they apply to registers.

7.3.1.1 UISA Registers

UISA registers are user-level registers that include the following.

7.3.1.1.1 General-Purpose Registers (GPRs)

The PowerPC architecture defines 32 user-level GPRs that are 32 bits wide in 32-bit cores. The GPRs serve as the data source or destination for all integer instructions.

7.3.1.1.2 Floating-Point Registers (FPRs)

The PowerPC architecture also defines 32 user-level, 64-bit FPRs. The FPRs serve as the data source or destination for floating-point instructions. These registers can contain data objects of either single- or double-precision floating-point formats. FPRs are not included in the e300c2 core.

7.3.1.1.3 Condition Register (CR)

The CR is a 32-bit user-level register that provides a mechanism for testing and branching. It consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point comparisons, arithmetic, and logical operations.

7.3.1.1.4 Floating-Point Status and Control Register (FPSCR)

The user-level FPSCR contains all floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE Std 754™. FPSCRs are not included in the e300c2 core.

7.3.1.1.5 User-Level SPRs

The PowerPC architecture defines numerous special purpose registers that serve a variety of functions, such as providing controls, indicating status, configuring the core, and performing special operations. During normal execution, a program can access the registers, as shown in Figure 7-2, depending on the program's access privilege (supervisor or user, determined by the privilege-level bit, MSR[PR]). Note that GPRs and FPRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions) or implicit, as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly. In the e300 core, all SPRs are 32 bits wide.

The following SPRs are accessible by user-level software:

- Link register (LR)—The LR can be used to provide the branch target address and to hold the return address after branch and link instructions. The LR is 32 bits wide in 32-bit implementations.
- Count register (CTR)—The CTR is decremented and tested automatically as a result of branch-and-count instructions. The CTR is 32 bits wide in 32-bit implementations.

- XER register—The 32-bit XER contains the summary overflow bit, integer carry bit, overflow bit, and a field specifying the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

7.3.1.2 VEA Registers

The VEA introduces the time base facility (TB) for reading. The TB is a 64-bit register pair whose contents are incremented once every four core input clock cycles. The TB consists of two 32-bit registers—time base upper (TBU) and time base lower (TBL). Note that the time base registers are read-only in user state.

7.3.1.3 OEA Registers

OEA registers are supervisor-level registers that include the following.

7.3.1.3.1 Machine State Register (MSR)

The MSR is a supervisor-level register that defines the state of the core. The contents of this register are saved when an interrupt is taken, and restored when the interrupt handling completes. A critical interrupt interrupt is taken in the e300 core when the *cint* signal is asserted and MSR[CE] is set. The e300 core implements the MSR as a 32-bit register.

Table 7-1 shows the bit definitions for MSR.

Table 7-1. MSR Bit Descriptions

Bits	Name	Description
0[1]	—	Reserved. Full function.
1–4[1]	—	Reserved. Partial function.
5–9[1]	—	Reserved. Full function.
10–12[1]	—	Reserved. Partial function.
13	POW	Power management enable (implementation-specific) 0 Disables programmable power modes (normal operation mode) 1 Enables programmable power modes (nap, doze, or sleep mode). This bit controls the programmable power modes only; it has no effect on dynamic power management (DPM). MSR[POW] may be altered with an **mtmsr** instruction only. Also, when altering the POW bit, software may alter only this bit in the MSR and no others. The **mtmsr** instruction must be followed by a context-synchronizing instruction.
14	TGPR	Temporary GPR remapping (implementation-specific) 0 Normal operation 1 TGPR mode. GPR0–GPR3 are remapped to TGPR0–TGPR3 for use by TLB miss routines. The contents of GPR0–GPR3 remain unchanged while MSR[TGPR] = 1. Attempts to use GPR4–GPR31 with MSR[TGPR] = 1 yield undefined results. Temporarily replaces TGPR0–TGPR3 with GPR0–GPR3 for use by TLB miss routines. The TGPR bit is set when either an instruction TLB miss, data read miss, or data write miss interrupt is taken. The TGPR bit is cleared by an **rfi** instruction.
15	ILE	Interrupt little-endian mode. When an interrupt occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the interrupt.

Table 7-1. MSR Bit Descriptions (continued)

Bits	Name	Description
16	EE	External interrupt enable 0 The processor ignores external interrupts, system management interrupts, and decrementer interrupts. 1 The processor is enabled to take an external interrupt, system management interrupt, or decrementer interrupt.
17	PR	Privilege level 0 The processor can execute both user- and supervisor-level instructions 1 The processor can only execute user-level instructions
18	FP	Floating-point available (Not supported on e300c2) 0 The processor prevents dispatch of floating-point instructions, including floating-point loads, stores, and moves. 1 The processor can execute floating-point instructions and can take floating-point enabled exception type program interrupts.
19	ME	Machine check enable 0 Machine check interrupts are disabled 1 Machine check interrupts are enabled
20	FE0	Floating-point exception mode 0 (Not supported on e300c2)
21	SE	Single-step trace enable 0 The processor executes instructions normally 1 The processor generates a trace interrupt upon the successful completion of the next instruction
22	BE	Branch trace enable 0 The processor executes branch instructions normally 1 The processor generates a trace interrupt upon the successful completion of a branch instruction
23	FE1	Floating-point exception mode 1 (Not supported on e300c2)
24	CE	Critical interrupt enable 0 Critical interrupts disabled 1 Critical interrupts enabled; critical interrupt and **rfci** instruction enabled The critical interrupt is an asynchronous implementation-specific interrupt. The critical interrupt vector offset is 0x00A00. The **rfci** instruction is implemented to return from these interrupt handlers. Also, CSRR0 and CSRR1 are used to save and restore the processor state for critical interrupts.
25	IP	Interrupt prefix. The setting of this bit specifies whether an interrupt vector offset is prepended with Fs or 0s. In the following description, *nnnnn* is the offset of the interrupt. 0 Interrupts are vectored to the physical address 0x000*n_nnnn* 1 Interrupts are vectored to the physical address 0xFFF*n_nnnn*
26	IR	Instruction address translation 0 Instruction address translation is disabled 1 Instruction address translation is enabled
27	DR	Data address translation 0 Data address translation is disabled 1 Data address translation is enabled
28–29[1]	—	Reserved. Full function.

Table 7-1. MSR Bit Descriptions (continued)

Bits	Name	Description
30	RI	Recoverable interrupt (for system reset and machine check interrupts) 0 Interrupt is not recoverable 1 Interrupt is recoverable
31	LE	Little-endian mode enable 0 The processor runs in big-endian mode 1 The processor runs in little-endian mode.

[1] All reserved bits should be set to zero for future compatibility.

7.3.1.3.2 Segment Registers (SRs)

For memory management, 32-bit processors implement sixteen 32-bit SRs. To speed access, the core implements the SRs as two arrays: a main array, for data memory accesses, and a shadow array, for instruction memory accesses. Loading a segment entry with the Move to Segment Register (**mtsr**) instruction loads both arrays.

7.3.1.3.3 Supervisor-Level SPRs

The e300 core, like the G2_LE core, has additional supervisor-level SPRs, which are shown in Figure 7-3. Two critical interrupt SPRs (CSRR0 and CSRR1), eight SPRGs (SPRG0–SPRG7), eight pairs of instruction BATs (IBAT0–IBAT7), eight pairs of data BATs (DBAT0–DBAT7), one system version register (SVR), one system memory base address (MBAR), one instruction address breakpoint control (IBCR), one data address breakpoint control (DBCR), a new instruction breakpoint register (IABR2), and two data address breakpoint registers (DABR and DABR2) are integrated into the core.

Supervisor-level SPRs include the following:

- The DSISR defines the cause of data access and alignment interrupts. The cause of a DSI interrupt for a data breakpoint (match with DABR and DABR2) can be determined by the value of the DSISR[DABR] bit (bit 9).
- The data address register (DAR) holds the address of an access after an alignment or DSI interrupt. For example, it contains the address of the breakpoint match condition.
- The decrementer register (DEC) is a 32-bit decrementing counter that provides a mechanism for causing a decrementer interrupt after a programmable delay.
- SDR1 specifies the page table format used in virtual-to-physical address translation for pages. (Note that physical address is referred to as ‘real address’ in the architecture specification.)
- The machine status save/restore register 0 (SRR0) is used for saving the address of the instruction that caused the interrupt, and the address to return to when a Return from Interrupt (**rfi**) instruction is executed.
- The machine status save/restore register 1 (SRR1) is used to save machine status on interrupts and to restore machine status when an **rfi** instruction is executed.
- The SPRG0–SPRG7 registers are provided for operating system use. They reduce the latency that may be incurred in the saving of registers to memory while in a handler. Note that the e300 implements four more SPRGs than the G2 (SPRG0–SPRG3).

- The time base register (TB) is a 64-bit register that maintains the time of day and operates interval timers. It consists of two 32-bit fields: time base upper (TBU) and time base lower (TBL).
- The processor version register (PVR) is a read-only register that identifies the version (model) and revision level of the processor. See Table 7-8 for the version and revision level of the PVR for the e300 processor core.
- Block address translation (BAT) arrays—The PowerPC architecture defines 16 BAT registers. The e300 core includes a total of eight pairs of DBAT and eight pairs of IBAT registers. See Figure 7-2 for a list of the SPR numbers for the BAT arrays.

The following supervisor-level SPRs are implementation-specific (not defined in the PowerPC architecture):

- DMISS and IMISS are read-only registers that are loaded automatically on an instruction or data TLB miss.
- HASH1 and HASH2 contain the physical addresses of the primary and secondary page table entry groups (PTEGs).
- ICMP and DCMP contain a duplicate of the first word in the page table entry (PTE) for which the table search is looking.
- The required physical address (RPA) register is loaded by the core with the second word of the correct PTE during a page table search.
- The system version register (SVR) is available on the e300 core, which identifies the specific version (model) and revision level of the system-on-a-chip (SOC) integration.
- System memory base address (MBAR) is an implementation-specific register available on the e300 core. It supports a temporary storage for the system-level memory map.
- The instruction and data address breakpoint registers (IABR, IABR2, DABR, DABR2) are loaded with an instruction or data address, respectively, that is compared to instruction addresses in the dispatch queue or to the data address in the LSU. When an address match occurs, a breakpoint interrupt is generated.
- One instruction breakpoint control register (IBCR) and one data breakpoint control register (DBCR) are implemented in the e300 core.
- To support critical interrupts, two registers (CSRR0 and CSRR1) are included in the e300 core.
- Eight SPRG registers (SPRG0–SPRG7) are in the e300 core.
- Block address translation (BAT) arrays—The e300 core has eight instruction and eight data BAT registers.
- The hardware implementation (HID0 and HID1) registers provide the means for enabling core checkstops and features and allow software to read the configuration of the PLL configuration signals. The HID2 register enables the true little-endian mode, cache way-locking, and the additional BAT registers.

Table 7-2 shows the bit definitions for HID0.

Table 7-2. e300 HID0 Bit Descriptions

Bits	Name	Function
0	EMCP	Enable *mcp*. The purpose of this bit is to mask out machine check interrupts caused by assertion of *mcp*, similar to how MSR[EE] can mask external interrupts. 0 Masks *mcp*. Asserting *mcp* does not generate a machine check interrupt or a checkstop. 1 Asserting *mcp* causes checkstop if MSR[ME] = 0 or a machine check interrupt if ME = 1
1	ECPE	Enable cache parity errors. 0 Disables instruction and data cache parity error reporting 1 Allows a detected cache parity error to cause a machine check interrupt if MSR[ME] = 1 or a checkstop if MSR[ME] = 0
2	EBA	Enable *ap_in[0:3]* and *ape* for address parity checking. 0 Disables address parity checking during a snoop operation 1 Allows an address parity error during snoop operations to cause a checkstop if MSR[ME] = 0 or a machine check interrupt if MSR[ME] = 1
3	EBD	Enable *dpe* for data parity checking. 0 Disables data parity checking 1 Allows a data parity error during reads to cause a checkstop if MSR[ME] = 0 or a machine check interrupt if MSR[ME] = 1
4	SBCLK	*clk_out* output enable. Used in conjunction with HID0[ECLK] and *hreset* to configure *clk_out.* See Table 7-3 for settings.
5	—	Reserved, should be cleared
6	ECLK	*clk_out* output enable. Used in conjunction with HID0[SBCLK] and the *hreset* signal to configure *clk_out.* See Table 7-3 for settings.
7	PAR	Disable precharge of *artry_out* 0 Precharge of *artry_out* enabled 1 Alters bus protocol slightly by preventing the processor from driving *artry_out* to high (negated) state. If this is done, the integrated device must restore the signals to the high state.
8	DOZE	Doze mode enable. Operates in conjunction with MSR[POW]. 0 Doze mode disabled 1 Doze mode enabled. Doze mode is invoked by setting MSR[POW] while this bit is set. In doze mode, the PLL, time base, and snooping remain active.
9	NAP	Nap mode enable. Operates in conjunction with MSR[POW]. 0 Nap mode disabled 1 Nap mode enabled. Nap mode is invoked by setting MSR[POW] while this bit is set. In nap mode, the PLL and time base remain active.
10	SLEEP	Sleep mode enable. Operates in conjunction with MSR[POW]. 0 Sleep mode disabled 1 Sleep mode enabled. Sleep mode is invoked by setting MSR[POW] while this bit is set. *qreq* is asserted to indicate that the processor is ready to enter sleep mode. If the system logic determines that the processor may enter sleep mode, the quiesce acknowledge signal, *qack,* is asserted back to the processor. Once *qack* assertion is detected, the processor enters sleep mode after several processor clocks. At this point, the system logic may turn off the PLL by first configuring *pll_cfg[0:6]* to PLL bypass mode, then disabling *sysclk*.

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
11	DPM	Dynamic power management enable 0 Dynamic power management is disabled 1 Functional units enter a low-power mode automatically if the unit is idle. This does not affect operational performance and is transparent to software or any external hardware.
12–15	—	Reserved, should be cleared.
16	ICE	Instruction cache enable 0 The instruction cache is neither accessed nor updated. All pages are accessed as if they were marked cache-inhibited (WIM = x1x). Potential cache accesses from the bus (snoop and cache operations) are ignored. In the disabled state for the L1 caches, the cache tag state bits are ignored and all instruction fetches are propagated to the coherent system bus (CSB) as single-beat transactions. For those transactions, however, $\overline{ci}$ reflects the state of the I bit in the MMU for that page regardless of cache disabled status. ICE is zero at power-up. 1 The instruction cache is enabled
17	DCE	Data cache enable 0 The data cache is neither accessed nor updated. All pages are accessed as if they were marked cache-inhibited (WIM = x1x). Potential cache accesses from the bus (snoop and cache operations) are ignored. In the disabled state for the L1 caches, the cache tag state bits are ignored and all data read and write accesses are propagated to the CSB as single-beat transactions. For those transactions, however, $\overline{ci}$ reflects the state of the I bit in the MMU for that page regardless of cache disabled status. DCE is zero at power-up. 1 The data cache is enabled
18	ILOCK	Instruction cache lock 0 Normal operation 1 The entire instruction cache is locked (that is, all eight ways of the cache are locked). A locked cache supplies data normally on a hit, but the access is treated as a cache-inhibited transaction on a miss. On a miss, the transaction to the bus is single-beat; however, $\overline{ci}$ still reflects the state of the I bit in the MMU for that page independent of cache locked or disabled status. To prevent locking during a cache access, an **isync** instruction must precede the setting of ILOCK.
19	DLOCK	Data cache lock 0 Normal operation 1 The entire data cache is locked (that is, all eight ways of the cache are locked). A locked cache supplies data normally on a hit, but is treated as a cache-inhibited transaction on a miss. On a miss, the transaction to the bus is single-beat; however, $\overline{ci}$ still reflects the state of the I bit in the MMU for that page independent of cache locked or disabled status. A snoop hit to a locked L1 data cache performs as if the cache were not locked. A cache block invalidated by a snoop remains invalid until the cache is unlocked. To prevent locking during a cache access, a **sync** instruction must precede the setting of DLOCK.
20	ICFI	Instruction cache Flash invalidate 0 The instruction cache is not invalidated. The bit is cleared when the invalidation operation begins (usually the next cycle after the write operation to the register). The instruction cache must be enabled for the invalidation to occur. 1 An invalidate operation is issued that marks the state of each instruction cache block as invalid. Cache access is blocked during this time. Setting ICFI clears all the valid bits of the blocks and the PLRU bits to point to way L0 of each set. For the e300 core, the proper use of the ICFI and DCFI bits is to set and clear them with two consecutive **mtspr** operations.

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
21	DCFI	Data cache Flash invalidate 0 The data cache is not invalidated. The bit is cleared when the invalidation operation begins (usually the next cycle after the write operation to the register). The data cache must be enabled for the invalidation to occur. 1 An invalidate operation is issued that marks the state of each data cache block as invalid without writing back modified cache blocks to memory. Cache access is blocked during this time. Bus accesses to the cache are signaled as a miss during invalidate-all operations. Setting DCFI clears all the valid bits of the blocks and the PLRU bits to point to way L0 of each set. For the e300 core, the proper use of the ICFI and DCFI bits is to set and clear them with two consecutive **mtspr** operations.
22–23	—	Reserved, should be cleared.
24	IFEM	Enable M bit on bus for instruction fetches 0 M bit not reflected on bus for instruction fetches. Instruction fetches are treated as nonglobal on the bus. 1 Instruction fetches reflect the M bit from the WIM settings
25	DECAREN	Decrementer auto reload 0 Normal operation. 1 Decrementer loads last mtdec value for precise periodic interrupt.
26	—	Reserved, should be cleared.
27	FBIOB	Force branch indirect on the bus 0 Register indirect branch targets are fetched normally 1 Forces register indirect branch targets to be fetched externally
28	ABE	Address broadcast enable. Controls whether certain address-only operations (such as cache operations) are broadcast on the bus. 0 Address-only operations affect only local caches and are not broadcast 1 Address-only operations are broadcast on the bus Affected instructions are **dcbi**, **dcbf**, and **dcbst**. Note that these cache control instruction broadcasts are not snooped by the e300 core. Refer to Section 4.3.3, “Data Cache Control,” for more information.
29–30	—	Reserved, should be cleared.
31	NOOPTI	No-op the data cache touch instructions 0 The **dcbt** and **dcbtst** instructions are enabled 1 The **dcbt** and **dcbtst** instructions are no-oped internal to the e300 core

Table 7-3 shows how HID0[ECLK] and HID0[SBCLK] are used to configure the *clk_out* signal.

Table 7-3. Using HID0[ECLK] and HID0[SBCLK] to Configure *clk_out*

hreset	**ECLK**	**SBCLK**	***clk_out***
Asserted	x	x	Bus clock (small pulse for every rising edge of sysclk)
Negated	0	0	Clock output off
	0	1	Core clock/2
	1	0	Core clock
	1	1	Bus clock

Table 7-4 shows the bit definitions for HID1

Table 7-4. HID1 Bit Descriptions

Bits	Name	Description
0	PC0	PLL configuration bit 0 (read-only)
1	PC1	PLL configuration bit 1 (read-only)
2	PC2	PLL configuration bit 2 (read-only)
3	PC3	PLL configuration bit 3 (read-only)
4	PC4	PLL configuration bit 4 (read-only)
5	PC5	PLL configuration bit 5 (read-only)
6	PC6	PLL configuration bit 6 (read-only)
7–31	—	Reserved, should be cleared

Note: The clock configuration bits reflect the state of the *pll_cfg[0:6]* signals.

Table 7-5 shows the bit definitions for HID2.

Table 7-5. e300HID2 Bit Descriptions

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	LET	True little-endian. This bit enables true little-endian mode operation for instruction and data accesses. This bit is set to reflect the state of the *tle* signal at the negation of $\overline{hreset}$. This bit is used in conjunction with MSR[LE] to determine the endian mode of operation. 0 No function 1 True little-endian mode, when MSR[LE] = 1 Changing the value of this bit during normal operation is not recommended
5	IFEB	Instruction fetch burst extension. This bit enables the instruction fetch burst extension. 0 Instruction fetch burst extension disabled 1 Instruction fetch burst extension enabled
6	—	Reserved, should be cleared.
7	MESISTATE	MESI state enable. This bit enables the four-state MESI cache coherency protocol. 0 MESI disabled. The data cache uses a three-state MEI coherency protocol. 1 MESI enabled. The data cache uses a four-state MESI protocol.
8	IFEC	Instruction fetch cancel extension. This bit enables the instruction fetch cancel extension. 0 Instruction fetch cancel extension disabled 1 Instruction fetch cancel extension enabled
9	EBQS	Enable BIU queue sharing. This bit enables data cache queue sharing. 0 Data cache queue sharing disabled 1 Data cache queue sharing enabled
10	EBPX	Enable BIU pipeline extension.This bit enables the bus interface unit pipeline extension. 0 BIU pipeline extension disabled; 1 level pipeline 1 BIU pipeline extension enabled; 1-1/2 level pipeline
11–12	—	Reserved for e300c1, should be cleared.

Table 7-5. e300HID2 Bit Descriptions (continued)

Bits	Name	Description
11	ELRW	Enable weighted LRU. This bit enables the use of an adjusted (weighted) LRU. 0 Normal operation. 1 The dcbt, dcbtst, and dcbz instructions use and adjusted (weighted) LRU such that they always select and replace the lowest unlocked way in the data cache.
12	NOKS	No kill for snoop. This bit enables the forcing of kill-type snoops to flush data instead of killing it. 0 Normal operation. 1 Forces write-with-kill snoops to flush instead of kill (snoop can never kill data).
13	HBE	High BAT enable. Regardless of the setting of HID2[HBE], these BATs are accessible by **mfspr** and **mtspr**. 0 IBAT[4–7] and DBAT[4–7] are disabled 1 IBAT[4–7] and DBAT[4–7] are enabled
14–15	—	Reserved, should be cleared.
16–18	IWLCK[0–2]	Instruction cache way-lock. Useful for locking blocks of instructions into the instruction cache for time-critical applications that require deterministic behavior. 000 no ways locked 001 way 0 locked 010 way 0 through way 1 locked 011 way 0 through way 2 locked 100 way 0 through way 2 locked in e300c2. 101 way 0 through way 2 locked in e300c2. 110 way 0 through way 2 locked in e300c2. 111 way 0 through way 2 locked in e300c2. Setting HID0[ILOCK] will lock all ways.
19	ICWP	Instruction cache way protection. Used to protect locked ways in the instruction cache from being invalidated. 0 Instruction cache way protection disabled 1 Instruction cache way protection enabled
20–23	—	Reserved, should be cleared.
24–26	DWLCK[0–2]	Data cache way-lock. Useful for locking blocks of data into the data cache for time-critical applications where deterministic behavior is required. 000 no ways locked 001 way 0 locked 010 way 0 through way 1 locked 011 way 0 through way 2 locked 100 way 0 through way 2 locked in e300c2. 101 way 0 through way 2 locked in e300c2. 110 way 0 through way 2 locked in e300c2. 111 way 0 through way 2 locked in e300c2. Setting HID0[DLOCK] will lock all ways.
27–31	—	Reserved, should be cleared.

7.3.2 Instruction Set and Addressing Modes

The following sections describe the PowerPC instruction set and addressing modes in general.

7.3.2.1 PowerPC Instruction Set and Addressing Modes

All PowerPC instructions are encoded as single-word (32-bit) opcodes. Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format simplifies instruction pipelining.

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include computational and logical instructions.
  - Integer arithmetic instructions
  - Integer compare instructions
  - Integer logical instructions
  - Integer rotate and shift instructions
- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
  - Floating-point arithmetic instructions
  - Floating-point multiply/add instructions
  - Floating-point rounding and conversion instructions
  - Floating-point compare instructions
  - Floating-point status and control instructions
- Load/store instructions—These include integer and floating-point load and store instructions.
  - Integer load and store instructions
  - Integer load and store multiple instructions
  - Floating-point load and store
  - Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)
- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  - Branch and trap instructions
  - Condition register logical instructions
- Processor control instructions—These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
  - Move to/from SPR instructions
  - Move to/from MSR
  - Synchronize
  - Instruction synchronize

- Memory control instructions—These instructions provide control of caches, TLBs, and segment registers.
  - Supervisor-level cache management instructions
  - Translation lookaside buffer management instructions. Note that there are additional implementation-specific instructions.
  - User-level cache instructions
  - Segment register manipulation instructions
- The e300 core implements the following instructions which are defined as optional by the PowerPC architecture:
  - Floating Select (**fsel**)
  - Floating Reciprocal Estimate Single-Precision (**fres**)
  - Floating Reciprocal Square Root Estimate (**frsqrte**)
  - Store Floating-Point as Integer Word (**stfiwx**)

Note that this grouping of instructions does not indicate the execution unit that executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are 4 bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for word and double-word operand loads and stores between memory and a set of 32 FPRs.

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

The core follows the program flow when it is in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of interrupt may cause one of several components of the system software to be invoked.

7.3.2.2 Implementation-Specific Instruction Set

The e300 core instruction set is defined as follows:

- The core provides hardware support for all 32-bit PowerPC instructions.
- The core provides two implementation-specific instructions used for software table search operations following TLB misses:
  - Load Data TLB Entry (**tlbld**)
  - Load Instruction TLB Entry (**tlbli**)
- The core implements the following instruction which is added to support critical interrupts (also supported on the G2_LE). This is a supervisor-level, context synchronizing instruction.
  - Return from Critical Interrupt (**rfci**)

- The core implements the following instruction which is added to support easy start-up initialization or reloading of the instruction cache.
  - Instruction Cache Block Touch (**icbt**)

7.3.3 Cache Implementation

The following sections describe the general cache characteristics as implemented in the PowerPC architecture and the core implementation.

7.3.3.1 PowerPC Cache Characteristics

The PowerPC architecture does not define hardware aspects of cache implementations. The e300 core controls the following memory access modes on a page or block basis:

- Write-back/write-through mode
- Caching-inhibited mode
- Memory coherency

Note that in the core, a cache block is defined as eight words. The VEA defines cache management instructions that provide a means by which the application programmer can affect the cache contents.

7.3.3.2 Implementation-Specific Cache Organization

The e300c2 provides 16-Kbyte, four-way set-associative instruction and data caches. The caches are physically addressed, and the data cache can operate in either write-back or write-through mode as specified by the PowerPC architecture.

The data cache is configured as 128 sets of 4 blocks each on the e300c2. Each block consists of 32 bytes, 2 state bits, and an address tag. The two state bits implement the three-state MEI (modified/exclusive/invalid) protocol. Each block contains eight 32-bit words. Note that the PowerPC architecture defines the term ‘block’ as the cacheable unit. For the core, the block size is equivalent to a cache line. A block diagram of the data cache organization is shown in Figure 7-3.

The instruction cache is configured as 128 sets of 4 blocks each on the e300c2. Each block consists of 32 bytes, an address tag, and a valid bit. The instruction cache may not be written to, except through a block fill operation. In the e300 core, the instruction cache is blocked only until the critical load completes. The e300 core supports instruction fetching from other instruction cache lines following the forwarding of the critical-first-double-word of a cache line load operation. Successive instruction fetches from the cache line being loaded are forwarded, and accesses to other instruction cache lines can proceed during the cache line load operation. The instruction cache is not snooped, and cache coherency must be maintained by software. A fast hardware invalidation capability is provided to support cache maintenance.

The e300c2data cache is configured as 128 sets of four blocks per set. The organization of the data cache is shown in Figure 7-3.

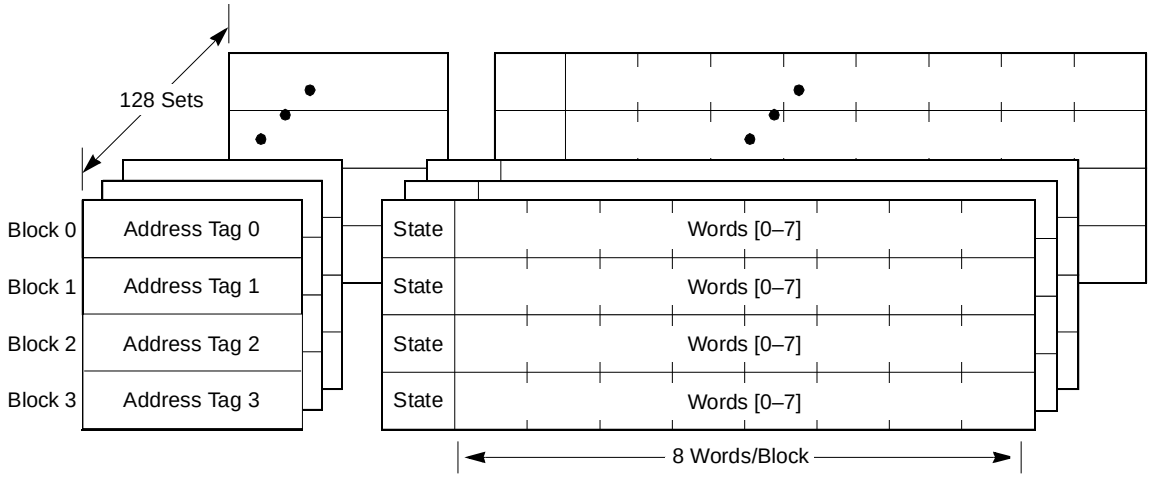


Figure 7-3. e300c2 and Data Cache Organization

Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits A[27–31] of the effective addresses are zero); thus, a cache block never crosses a page boundary. Misaligned accesses across a page boundary can incur a performance penalty.

The e300 core cache blocks are loaded in four beats of 64 bits each on the 64-bit data bus. The burst load is performed as critical-double-word-first. The data cache is blocked to internal accesses until the load completes; the instruction cache allows sequential fetching during a cache block load. In the core, the critical-double-word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.

To ensure coherency among caches in a multiprocessor (or multiple caching-device) implementation, the core implements the MEI protocol during normal operation of the data cache. The new data cache MESI extension supports the additional fourth cache coherency shared state for the data cache. To support this feature, the shared signal, *shd*, has been added to the bus interface. Although the MESI protocol is supported by the e300 core, it is not implemented on MPC8323E. The following four states indicate the state of the cache block:

- Modified—The cache block is modified with respect to system memory; that is, data for this address is valid only in the cache and not in system memory.
- Exclusive—This cache block holds valid data that is identical to the data at this address in system memory. No other cache has this data.
- Shared—Only available if HID2[MESISTATE] register bit is set. The address block is valid in the cache and in at least one other cache. This block is always consistent with system memory. That is, the shared state is shared-unmodified; there is no shared-modified state. Although the MESI protocol is supported by the e300 core, it is not implemented on MPC8323E.
- Invalid—This cache block does not hold valid data.

Cache coherency is enforced by on-chip bus snooping logic. Because the e300 core data cache tags are single-ported, a simultaneous load/store and snoop access represents a resource contention. The snoop access is given first access to the tags. The load or store then occurs on the clock following the snoop.

Parity is now integrated into both instruction and data cache memory. A machine check interrupt is now taken upon the detection of an instruction or data cache parity error. Parity is checked whenever valid data is returned from the instruction or data cache for a cache hit or whenever valid data is read out of the cache for a castout or snoop-push operation.

7.3.3.3 Instruction and Data Cache Way-Locking

The e300 core implements instruction and data cache way-locking, which guarantees that certain memory accesses will hit in the cache. This provides deterministic access times for those accesses.

7.3.4 Interrupt Model

This section describes the PowerPC interrupt model and the e300 core implementation specifically.

7.3.4.1 PowerPC Interrupt Model

The PowerPC interrupt mechanism allows the core to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. The conditions that can cause interrupts are called exceptions. When interrupts occur, information about the state of the core is saved to certain registers and the core begins execution at an address (interrupt vector) predetermined for each interrupt type. Interrupts are processed in supervisor mode.

Some interrupts, such as program interrupts, can be triggered by a broad range of exception conditions. Other interrupts, such as the decrementer interrupt, have only a single exception condition. Although multiple exception conditions can map to a single interrupt vector, a more specific condition may be determined by examining a register associated with the interrupt—for example, the DSISR and the FPSCR. Additionally, some exception conditions can be explicitly enabled or disabled by software.

The PowerPC architecture requires that interrupts be handled in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are presented strictly in order. When an instruction-caused interrupt is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that have not yet entered the execute stage, are required to complete before the interrupt is taken. Any interrupts caused by those instructions are handled first. Likewise, asynchronous, precise interrupts are recognized when they occur, but are not handled until the instruction currently in the completion stage successfully completes execution or generates an interrupt, and the completed store queue is emptied.

Unless a catastrophic condition causes a system reset or machine check interrupt, only one interrupt is handled at a time. If, for example, a single instruction encounters multiple interrupt conditions, those conditions are handled sequentially. After the interrupt handler completes, the instruction execution continues until the next interrupt condition is encountered. However, in many cases there is no attempt to re-execute the instruction. This method of recognizing and handling interrupts sequentially guarantees that interrupts are recoverable.

To prevent the program state from being lost due to a system reset, a machine check interrupt, or an instruction-caused interrupt in the interrupt handler, interrupt handlers should save the information stored in SRR0 and SRR1 early and before enabling external interrupts.

The PowerPC architecture supports four types of interrupts:

- Synchronous, precise—These are caused by instructions. All instruction-caused interrupts are handled precisely; that is, the machine state at the time the interrupt occurs is known and can be completely restored. This means that (excluding the trap and system call interrupts) the address of the faulting instruction is provided to the interrupt handler and neither the faulting instruction nor subsequent instructions in the code stream will complete execution before the interrupt is taken. Once the interrupt is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the interrupt handler). When an interrupt is taken due to a trap or system call instruction, execution resumes at an address provided by the handler.
- Synchronous, imprecise—The PowerPC architecture defines two imprecise floating-point exception modes: recoverable and nonrecoverable. Even though the core provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, all enabled floating-point exceptions are always precise on the core).
- Asynchronous, maskable—The external system management interrupt (SMI) and decrementer interrupts are maskable, asynchronous interrupts. When these interrupts occur, their handling is postponed until the next instruction and any of its associated interrupts complete execution. If there are no instructions in the execution units, the interrupt is taken immediately upon determination of the correct restart address (for loading SRR0).
- Asynchronous, nonmaskable—The system reset and the machine check interrupt are nonmaskable, asynchronous interrupts. They may not be recoverable, or they may provide a limited degree of recoverability. All interrupts report recoverability through MSR[RI].

7.3.4.2 Implementation-Specific Interrupt Model

As specified by the PowerPC architecture, all interrupts can be described as either precise or imprecise and either synchronous or asynchronous. Asynchronous interrupts (some of which are maskable) are caused by events external to the processor's execution; synchronous interrupts, which are all handled precisely by the e300 core, are caused by instructions. A system management interrupt is an implementation-specific interrupt. The interrupt classes are shown in Table 7-6.

Table 7-6. Interrupt Classifications

Synchronous/Asynchronous	Precise/Imprecise	Interrupt Type
Asynchronous, nonmaskable	Imprecise	Machine check System reset
Asynchronous, maskable	Precise	External interrupt Decrementer System management interrupt Critical interrupt
Synchronous	Precise	Instruction-caused interrupts

Although interrupts have other characteristics, such as whether they are maskable, the distinctions shown in Table 7-6 define categories of interrupts that the core handles uniquely. Note that Table 7-6 includes no synchronous, imprecise instructions. While the PowerPC architecture supports imprecise handling of floating-point exceptions, the core implements floating-point exception modes as precise.

The e300 core interrupts and exception conditions that cause them are listed in Table 7-7.

Table 7-7. Exceptions and Interrupts

Interrupt Type	Vector Offset (hex)	Exception Conditions
Reserved	00000	—
System reset	00100	Caused by the assertion of either *hreset*.
Machine check	00200	Caused by the assertion of the *tea* signal during a data bus transaction, assertion of *mcp*, an address or data parity error, or an instruction or data cache parity error. Note that the e300 has SRR1 register values that are different from the G2/G2_LE cores' when a machine check occurs.
DSI	00300	Determined by the bit settings in the DSISR, listed as follows: 1 Set if the translation of an attempted access is not found in the primary hash table entry group (HTEG), or in the rehashed secondary HTEG, or in the range of a DBAT register; otherwise cleared 4 Set if a memory access is not permitted by the page or DBAT protection mechanism; otherwise cleared 6 Set for a store operation and cleared for a load operation 9 Set if a data address breakpoint interrupt occurs when the data [0–28] in the DABR or DABR2 matches the next data access (load or store instruction) to complete in the completion unit. The different breakpoints are enabled as follows: • Write breakpoints enabled when DABR[30] is set • Read breakpoints enabled when DABR[31] is set
ISI	00400	Caused when an instruction fetch cannot be performed for any of the following reasons: • The effective (logical) address cannot be translated. That is, there is a page fault for this portion of the translation, so an ISI interrupt must be taken to load the PTE (and possibly the page) into memory. • The fetch access violates memory protection (indicated by SRR1[4] set). If the key bits (Ks and Kp) in the segment register and the PP bits in the PTE are set to prohibit read access, instructions cannot be fetched from this location.
External interrupt	00500	Caused when MSR[EE] = 1 and the *int* signal is asserted.
Alignment	00600	Caused when the core cannot perform a memory access for any of the reasons described below: • The operand of a floating-point load or store instruction is not word-aligned. • The operands of **lmw**, **stmw**, **lwarx**, and **stwcx.** instructions are not aligned. • The instruction is **lswi**, **lswx**, **stswi**, **stswx**, and the core is in little-endian mode. Note that PowerPC little-endian mode is not supported on the e300 core. • The operand of **dcbz** is in memory that is write-through-required or caching-inhibited.

Table 7-7. Exceptions and Interrupts (continued)

Interrupt Type	Vector Offset (hex)	Exception Conditions
Program	00700	Caused by one of the following exception conditions, which correspond to bit settings in SRR1 and arise during execution of an instruction. Floating-point enabled exception—A floating-point enabled exception condition is generated when the following condition is met: (MSR[FE0] | MSR[FE1]) and FPSCR[FEX] is 1. • FPSCR[FEX] is set by the execution of a floating-point instruction that causes an enabled exception or by the execution of one of the Move to FPSCR instructions that results in both an exception condition bit and its corresponding enable bit being set in the FPSCR. • Illegal instruction—An illegal instruction program interrupt is generated when execution of an instruction is attempted with an illegal opcode or illegal combination of opcode and extended opcode fields (including PowerPC instructions not implemented in the core), or when execution of an optional instruction not provided in the core is attempted (these do not include those optional instructions that are treated as no-ops). • Privileged instruction—A privileged instruction program interrupt is generated when the execution of a privileged instruction is attempted and the MSR register user privilege bit, MSR[PR], is set. In the e300 core, this interrupt is generated for **mtspr** or **mfspr** with an invalid SPR field if SPR[0] = 1 and MSR[PR] = 1. This may not be true for all cores that implement the PowerPC architecture. • Trap—A trap type program interrupt is generated when any of the conditions specified in a trap instruction are met.
Floating-point unavailable	00800	Caused by an attempt to execute a floating-point instruction (including floating-point load, store, and move instructions) when the floating-point available bit (MSR[FP]) is cleared. In the e300c2 core, any attempt to execute a floating-point instruction results in a floating-point unavailable exception.
Decrementer	00900	Occurs when DEC[0] changes from 0 to 1. This interrupt is enabled with MSR[EE].
Critical interrupt	00A00	Taken when $\overline{cint}$ is asserted and MSR[CE] = 1.
Reserved	00B00–00BFF	—
System call	00C00	Occurs when a System Call (**sc**) instruction is executed.
Trace	00D00	Taken when MSR[SE] =1 or when the currently completing instruction is a branch and MSR[BE] =1.
Reserved	00E00	The e300 core does not generate an interrupt to this vector. Other devices may use this vector for floating-point assist interrupts.
Instruction translation miss	01000	Caused when the effective address for an instruction fetch cannot be translated by the ITLB.
Data load translation miss	01100	Caused when the effective address for a data load operation cannot be translated by the DTLB.
Data store translation miss	01200	Caused when the effective address for a data store operation cannot be translated by the DTLB, or when a DTLB hit occurs and the change bit in the PTE must be set due to a data store operation.
Instruction address breakpoint	01300	Occurs when the address (bits 0–29) in the IABR matches the next instruction to complete in the completion unit, and IABR[30] is set. Note that the e300 core also implements IABR2, which functions identically to IABR.

Table 7-7. Exceptions and Interrupts (continued)

Interrupt Type	Vector Offset (hex)	Exception Conditions
System management interrupt	01400	Caused when MSR[EE] = 1 and the $\overline{smi}$ input signal is asserted.
Reserved	01500–02FFF	—

7.3.5 Memory Management

The following sections describe the memory management features of the PowerPC architecture and the e300 core implementation, respectively.

7.3.5.1 PowerPC Memory Management

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses and to provide access protection on blocks and pages of memory.

The core generates two types of accesses that require address translation: instruction accesses and data accesses to memory generated by load and store instructions.

The PowerPC MMU and interrupt model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged implies that individual pages are loaded into physical memory from system memory only when they are first accessed by an executing program.

The hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

The page table contains a number of page-table entry groups (PTEGs). A PTEG contains eight page-table entries (PTEs) of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

Address translations are enabled by setting bits in the MSR—MSR[IR] enables instruction address translations, and MSR[DR] enables data address translations.

7.3.5.2 Implementation-Specific Memory Management

The instruction and data memory management units in the e300 core provide 4 Gbytes of logical address space accessible to supervisor and user programs with a 4-Kbyte page size and 256-Mbyte segment size. Block sizes range from 128 Kbytes to 256 Mbytes and are software selectable. In addition, the core uses an interim 52-bit virtual address and hashed page tables for generating 32-bit physical addresses. The MMUs in the e300 core rely on the interrupt processing mechanism for the implementation of the paged virtual memory environment and for enforcing protection of designated memory areas.

Instruction and data TLBs provide address translation in parallel with the on-chip cache access, incurring no additional time penalty in the event of a TLB hit. A TLB is a cache of the most recently used page table

entries. Software is responsible for maintaining the consistency of the TLB with memory. The core TLBs are 64-entry, two-way, set-associative caches that contain instruction and data address translations. The core provides hardware assist for software table search operations through the hashed page table on TLB misses. Supervisor software can invalidate TLB entries selectively.

For instructions and data that correspond to block address translation, the e300 core provides independent eight-entry BAT arrays. These entries define blocks that can vary from 128 Kbytes to 256 Mbytes. The BAT arrays are maintained by system software. HID2[HBE] is added to the e300 for enabling or disabling the four additional pairs of BAT registers. However, regardless of the setting of HID2[HBE], these BATs are accessible by **mfspr** and **mtspr**.

As specified by the PowerPC architecture, the hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

Also as specified by the PowerPC architecture, the page table contains a number of PTEGs. A PTEG contains 8 PTEs of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

7.3.6 Instruction Timing

The e300 core is a pipelined superscalar processor core. Because instruction processing is reduced into a series of stages, an instruction does not require all of the resources of an execution unit at the same time. For example, after an instruction completes the decode stage, it can pass on to the next stage, while the subsequent instruction can advance into the decode stage. This improves the throughput of the instruction flow. For example, it may take three cycles for a single floating-point instruction to execute, but if there are no stalls in the floating-point pipeline, a series of floating-point instructions can have a throughput of one instruction per cycle.

The core instruction pipeline has four major pipeline stages, described as follows:

- The fetch pipeline stage primarily involves retrieving instructions from the memory system and determining the location of the next instruction fetch. Additionally, if possible, the BPU decodes branches during the fetch stage and folds out branch instructions before the dispatch stage.
- The dispatch pipeline stage is responsible for decoding the instructions supplied by the instruction fetch stage and determining which of the instructions are eligible to be dispatched in the current cycle. In addition, the source operands of the instructions are read from the appropriate register file and dispatched with the instruction to the execute pipeline stage. At the end of the dispatch pipeline stage, the dispatched instructions and their operands are latched by the appropriate execution unit.
- In the execute pipeline stage, each execution unit with an instruction executes the selected instruction (perhaps over multiple cycles), writes the instruction's result into the appropriate rename register, and notifies the completion stage when the execution has finished. In the case of an internal interrupt, the execution unit reports the interrupt to the completion/write-back pipeline stage and discontinues instruction execution until the interrupt is handled. The interrupt is not signaled until that instruction is the next to be completed. Execution of most floating-point instructions is pipelined within the FPU, allowing up to three instructions to execute in the FPU concurrently. The FPU pipeline stages are multiply, add, and round-convert. The LSU has two

pipeline stages: the first stage, for effective address calculation and MMU translation, and the second, for accessing data in the cache.

- The complete/write-back pipeline stage maintains the correct architectural machine state and transfers the contents of the rename registers to the GPRs and FPRs as instructions are retired. If the completion logic detects an instruction causing an interrupt, all subsequent instructions are canceled, their execution results in rename registers are discarded, and instructions are fetched from the correct instruction stream.

A superscalar processor core issues multiple, independent instructions into multiple pipelines, allowing instructions to execute in parallel. The e300c1 core has independent execution units for: integer instructions, branch instructions, load/store instructions, and system register instructions.The e300c2 does not include a floating-point unit. The e300c2 provides two IUs, which improves the throughput of integer instructions. The e300c2 provides two integer units for greater integer instruction throughput along with enhanced multipliers in each IU that reduce the multiply instruction latency to a maximum of two cycles. The IU have dedicated register files for maintaining operands (GPRs), allowing integer calculations to occur simultaneously without interference. The e300c2 does not include floating-point registers.

The core provides support for single-cycle store, and it provides an adder/comparator in the system register unit that allows the dispatch and execution of multiple integer add and compare instructions on each cycle.

Because the PowerPC architecture can be applied to such a wide variety of implementations, instruction timing among processor cores varies accordingly.

7.3.7 Core Interface

The core interface is specific for each processor core implementation.

The MPC8323E contains an internal coherent system bus (CSB) that interfaces the processor core to the peripheral logic. This internal bus is very similar in function to the external 60x bus interface on the MPC603e. In the case of the MPC8323E, the CSB system logic decodes e300-initiated transactions and directs all accesses to the appropriate interface.

The e300 core can operate at a variety of frequencies allowing the designer to trade off performance for power consumption. The processor core is clocked from a separate PLL, which is referenced to the CSB frequency. This allows the processor core and the peripheral logic to operate at different frequencies.

The e300 core provides a versatile core interface that allows for a wide range of implementations. The interface includes a 32-bit address bus, a 64-bit data bus, and 56 control and information signals (see Figure 7-4). The core interface allows for address-only transactions, as well as address and data transactions. The core control and information signals include the address arbitration, address start, address transfer, transfer attribute, address termination, data arbitration, data transfer, data termination, and core state signals. Test and control signals provide diagnostics for selected internal circuits.

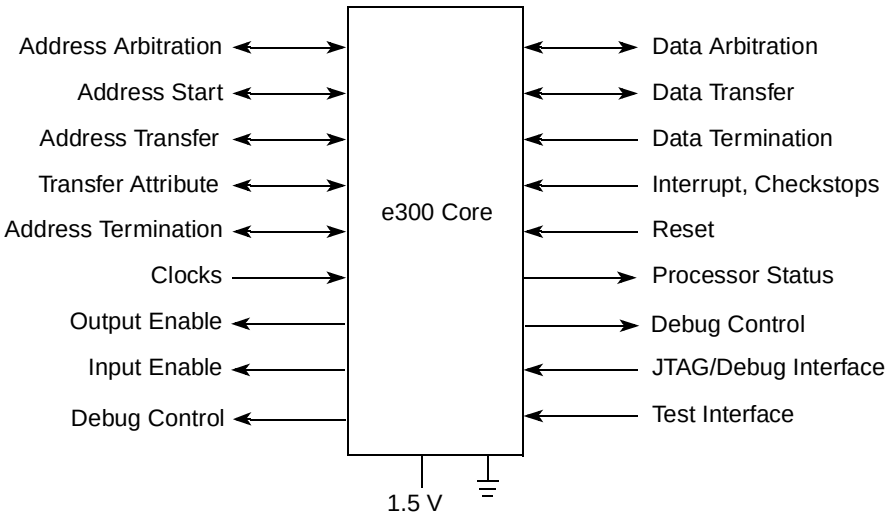


Figure 7-4. Core Interface

The core interface supports bus pipelining, allowing the address tenure of one transaction to overlap the data tenure of another. The extent of the pipelining depends on external arbitration and control circuitry. Similarly, the core supports split-bus transactions for systems with multiple potential bus masters—one device can have mastership of the address bus while another has mastership of the data bus. Allowing multiple bus transactions to occur simultaneously increases the available bus bandwidth for other activity and, as a result, improves performance.

The core clocking structure allows the bus to operate at integer multiples of the core cycle time.

The following sections describe the core bus support for memory operations. Note that some signals perform different functions depending on the addressing protocol used.

7.3.7.1 Memory Accesses

The e300 core CSB is a 64-bit data bus.

With a 64-bit CSB, memory accesses allow transfer sizes of 8, 16, 24, 32, 40, 48, 56, or 64 bits in one bus clock cycle. Data transfers occur in either single-beat transactions or four-beat burst transactions. Single-beat transactions are caused by noncached accesses that access memory directly (that is, reads and writes when caching is disabled, caching-inhibited accesses, and stores in write-through mode). Four-beat burst transactions, which always transfer an entire cache block (32 bytes), are initiated when a line is read from or written to memory.

7.3.7.2 Signals

The e300 core signals are grouped as follows:

- Interrupts/Resets—These signals include the external interrupt signal ($\overline{int}$), critical interrupt signal ($\overline{cint}$), checkstop signals, and both soft reset and hard reset signals. They are used to interrupt and, under various conditions, to reset the core.

- JTAG/debug interface signals—The JTAG (based on IEEE 1149.1) interface and debug unit provides a serial interface to the system for performing monitoring and boundary tests. Two additional signals are added to the e300 core to allow observation of the internal clock state of the core (*stopped*) and to allow the external input to force the core into a halted state (*ext_halt*).
- Core status and control—These signals include the memory reservation signal, machine quiesce control signals, time base/decrementer clock base enable signal, and the *tlbisync* signal.
- Clock control—These signals provide for system clock input and frequency control.
- Test interface signals—Signals like address matching, combinational matching, and watchpoint are used in the core for production testing.
- Transfer attribute signals—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is bursted, write-through, or cache-inhibited.

7.3.8 Debug Features

Some new debug features are specific to the e300 core. Accesses to the debug facilities are available only in supervisor mode by using the **mtspr** and **mfspr** instructions. The e300 provides the following additional feature in the JTAG/debug interface: Inclusion of breakpoint status and control pins: *stopped* and *ext_halt.*

7.3.8.1 Breakpoint Signaling

The breakpoint signaling provided on the e300 core allows observability of breakpoint matches external to the core. The *iabr, iabr2, dabr,* and *dabr2* breakpoint signals are asserted for at least one bus clock cycle when the respective breakpoint occurs. The status of the run state of the e300 core is indicated by the *stopped* pin. An asynchronous external breakpoint can be asserted to the e300 core using the *ext_halt* pin:

- When DBCR and IBCR are configured for an OR combinational signal type, the breakpoint signals *iabr, iabr2* and *dabr, dabr2* reflect their respective breakpoints.
- When the DBCR and IBCR are configured for AND combinational signal type, only the *iabr2* and *dabr2* breakpoint signals are asserted after the AND condition is met (that is, both instruction breakpoints occurred or both data breakpoints occurred).
- When the core_stopped pin is asserted, the e300 core has entered a stopped state and all internal clocking has stopped, indicating that a hardware debug event has occurred.
- The *ext_halt* input pin can be used to force the core into halted state. The halted state may be a hardstop, conditional upon the HARDSTOP condition being set through the JTAG/debug interface

7.4 Differences Between Cores

The e300 core has similar functionality to the G2_LE core. Table 7-8 describes the differences between the G2_LE and the e300.

Table 7-8. Differences Between e300 and G2_LE Cores

e300 Core	G2_LE Core	Impact
New HID0 bits	—	The e300 core has a new HID0 bit defined to enable cache parity error reporting (ECPE).
New HID1 bits	—	The e300 core has new HID1 bits defined to extend the number of PLL configuration signals to seven (PC5, PC6).
New HID2 bits	—	The e300 core has new HID2 bits defined to support instruction fetch bursting (IFEB), MESI coherency protocol (MESI), instruction fetch cancels (IFEC), data cache queue sharing (EBQS), pipelining extension (EBPX), additional cache way locking (IWLCK and DWLCK), and instruction cache way protection (ICWP).
New PVR register value	—	The processor version register values differ.
New IBCR and DBCR bits	—	The e300 core has new IBCR[IABRSTAT, IABR2STAT] and DBCR[DABR1STAT, DABR2STAT] fields to provide instruction and data address breakpoint status.
—	16-Kbyte, four-way, set-associative, instruction and data caches	Some e300 cores may have different cache sizes than the G2_LE. See the *e300 PowerPC Core Reference Manual* for detailed information.
L1 cache parity	—	The e300 core supports parity for both instruction and data caches; the G2_LE does not support cache parity.
MEI or MESI coherency protocols	MEI protocol only	The e300 supports two coherency protocols: MEI and MESI; the G2_LE only supports the MEI protocol.Not implemented on MPC8323E.
Instruction cancel extension	—	The e300 instruction cancel mechanism improves utilization of instruction cache by supporting ‘hits-under-cancels’ and ‘misses-under-cancels’; the G2_LE requires the cancel to complete before new instruction fetches can begin.
Instruction fetch bursts to caching-inhibited space	Single-beat instruction fetches to caching-inhibited space	The e300’s instruction fetch burst extension allows all caching-inhibited instruction fetches to be performed on the bus as burst transactions, even though the instructions are not cached. This improves performance for instruction space that is caching-inhibited, because up to eight instructions are returned with one bus operation. The G2_LE core must use single-beat instruction fetches for caching-inhibited space, returning only two instructions per bus operation.
Instruction cache way protection	—	The e300 core can protect locked ways in the instruction cache from invalidation; the G2_LE does not support instruction cache way protection.

Table 7-8. Differences Between e300 and G2_LE Cores (continued)

e300 Core	G2_LE Core	Impact
Data cache queue sharing	—	The e300 has a new data cache queue sharing extension that allows the two burst-write queues in the bus unit to be used interchangeably for cache replacements and snoop pushes. Thus, the data cache can support two outstanding cache replacements or two outstanding snoop push operations on the bus at any given time.
icbt instruction	—	The e300 supports a new instruction cache block touch instruction that facilitates preloading the instruction cache before locking; the G2_LE core requires speculatively fetching instructions before locking the instruction cache.
1-1/2-level bus pipelining	1-level bus pipelining	For the e300, a new transaction can complete an address tenure when the previous transaction has been granted the data bus; for the G2_LE, a new transaction must wait until the previous data tenure has completed before completing its address tenure.
PowerPC little-endian not supported	PowerPC little-endian supported	PowerPC little-endian will not be supported in the e300 core, although true little-endian will be fully supported.
Data retry mode removed	Data retry mode available	$\overline{drtry}$ and *drtrymode* will no longer be supported on the e300 and future versions.
External control instructions removed	External control instructions available	The **eciwx** and **ecowx** instruction pair will not be supported on the e300 core. These are optional instructions in the PowerPC architecture.
Reduced pin mode removed	Reduced pin mode available	Reduced pinout mode and the signal *redpinmode* will not be supported in the e300 core.

Chapter 8
Integrated Programmable Interrupt Controller (IPIC)

This chapter describes the integrated programmable interrupt controller (IPIC), including a definition of the external signals and their functions. Also, the configuration, control, and status registers are described in this chapter. Note that individual chapters in this reference manual describe specific initialization aspects for each individual block.

8.1 Introduction

This chapter describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. The programming model is similar to the interrupt controller of the MPC8260. The interrupt controller provides interrupt management that is responsible for receiving hardware-generated interrupts from different sources (both internal and external). It also prioritizes and delivers the interrupts to the CPU for servicing. The IPIC prioritizes and manages interrupts from the following controller units:

- DDR memory controller (DDR)
- Local bus memory controller (LBC)
- PCI
- Four-channel DMA controller (DMA)
- Message unit (MU)
- DUART communication module (DUART)
- Security engine (SEC)
- System bus arbiter (SBA)
- Periodic interval timer (PIT)
- Real time clock timer (RTC ALR and RTC SEC)
- Eight global timers (GTM1–GTM8)
- Software watchdog timer (WDT)
- I^2C controller (I^2C)
- Power management controller (PMC)
- QUICC Engine block
- External pins ($\overline{\text{IRQ}}$[0:7])

The interrupt sources controlled by the IPIC unit cause exceptions in the processor core. The internal interrupt ($\overline{int}$) signal is the main interrupt output from the IPIC to the core and it causes the regular interrupt exception. The $\overline{cint}$ signal is the critical interrupt output from the IPIC to the processor core and causes the critical interrupt exception. The $\overline{\text{smi}}$ signal is the system management interrupt output from the IPIC to the processor core and causes the system management interrupt exception. The machine check exception is

caused by the internal $\overline{mcp}$ signal generated by the IPIC, informing the host processor of error conditions, assertion of the external $\overline{IRQ0}$ machine-check request (enabled when SEMSR[SIRQ0] = 1), and other conditions.

Table 8-1 shows the relationship of the various functional blocks and external signals of the device to the IPIC unit.

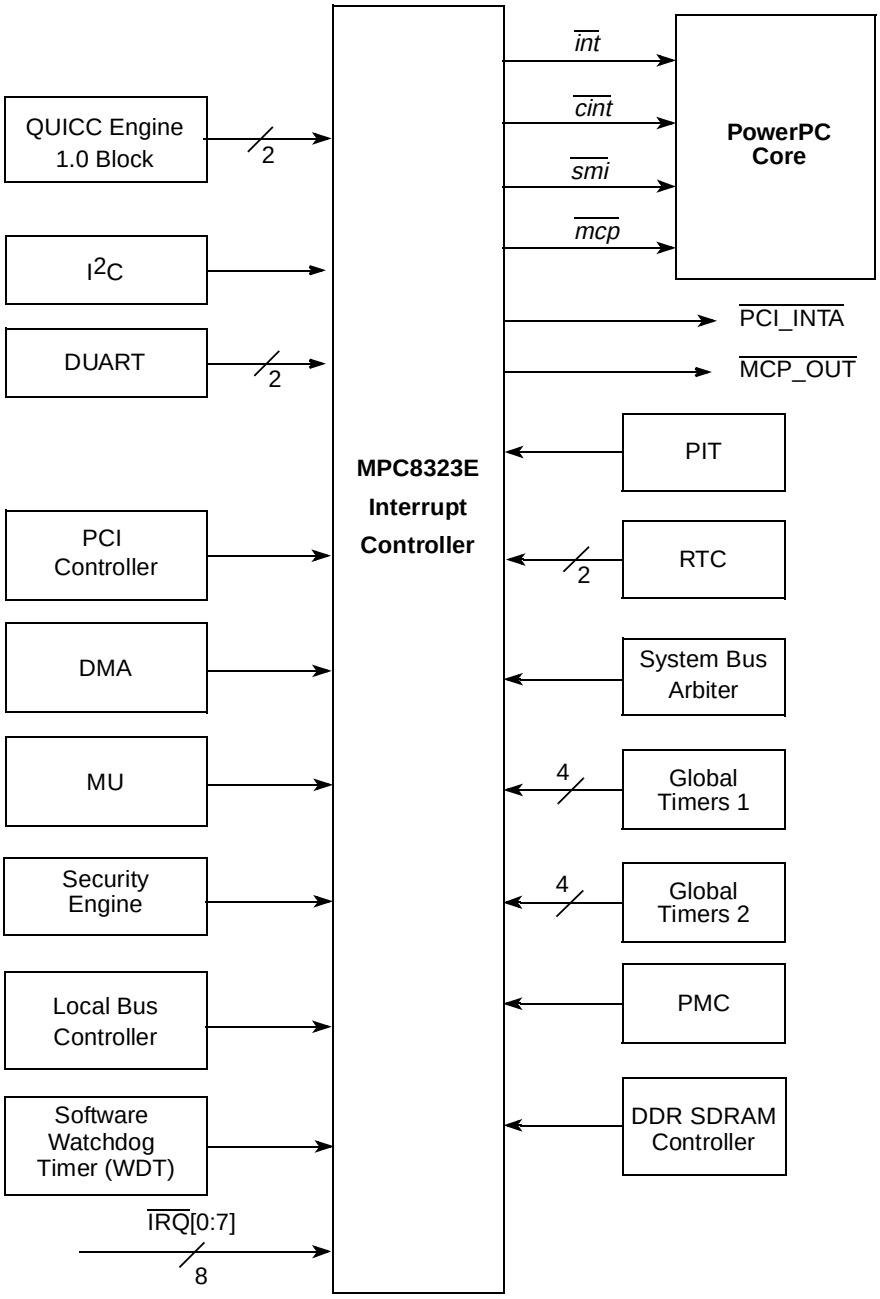
The IPIC receives interrupt request signals from the following two sources:

- External to the integrated device
- Internal to the integrated device

The unit selects the highest priority interrupt from all current interrupts and forwards it to the internal processor core, or off-chip for external servicing.

The IPIC also manages an internal non-maskable machine-check processor ($\overline{mcp}$) signal and the interrupt generated by the off-chip interrupt sources ($\overline{IRQ}$[0:7]).

The interrupt router of the IPIC monitors the outputs of the internal configuration registers. When the priority is highest in one of the received interrupt signals, the IPIC sets the corresponding bit in one of the interrupt pending registers—system internal interrupt pending register (SIPNR)/system external interrupt pending register (SEPNR). If the interrupt is not masked, the IPIC asserts the $\overline{int}$ signal to indicate an interrupt request to the processor. When the processor is running the specific $\overline{int}$, $\overline{cint}$, or $\overline{smi}$ interrupt handler code, the processor must vectorize the external interrupt handler by explicitly (in software) reading the corresponding interrupt vector register (SIVCR, SCVCR or SMVCR). In response to this read, the IPIC unit returns the vector (associated with the interrupt source) to the interrupt handler routine. In addition, the handler can vectorize different branches of interrupt handling.



The IPIC receives the following types of interrupts:

- External interrupt—triggered by the off-chip signals ($\overline{\text{IRQ}}n$) listed in Table 8-1
- Internal interrupts—on-chip interrupts, triggered by the sources listed in Table 8-8 and Table 8-10
- External and internal non-maskable machine check conditions, signaled by the sources listed in Table 8-22 through $\overline{mcp}$

The interrupt controller provides the ability to mask each interrupt source. Any source that can be caused by multiple events are also maskable.

When the IPIC receives an internal or external interrupt, its configuration register is checked to determine if it should be routed off-chip (to the external $\overline{\text{PCI\_INTA}}$) or serviced as a normal external interrupt by the processor core (through the $\overline{int}$ signal). As a third alternative, if the incoming interrupt has been configured as a critical or system management interrupt, the IPIC completes the processing of the interrupt by asserting $\overline{cint}$ or $\overline{smi}$ to the core. The assertion of the $\overline{cint}$ or $\overline{smi}$ signal to the core causes the interrupt to be serviced as a critical or a system management interrupt, respectively.

8.2 Features

The IPIC unit implements the following features:

- Functional and programming compatibility with the MPC8260 interrupt controller
- Support for external and internal discrete vectorized interrupt sources
- Support for external and internal non-maskable machine check conditions, signaled by $\overline{mcp}$
- Support for dedicated external interrupts—QUICC Engine ports interrupts
- Programmable highest priority request (can be programmed to support a critical ($\overline{cint}$) or system management interrupt ($\overline{smi}$) type)
- Two programmable priority mixed groups of four on-chip and four external interrupt signals with two priority schemes for each group: grouped and spread
- Two programmable priority internal groups of eight on-chip interrupt signals with two priority schemes for each group: grouped and spread
- Two highest priority interrupts from each group can be programmed to support a critical or system management interrupt type
- External and internal interrupts directed to host processor
- Unique vector number for each interrupt source

8.3 Modes of Operation

The IPIC unit can operate in the core enable or core disable mode.

8.3.1 Core Enable Mode

In core enable mode, all internal interrupts (including those from the PCI block) are routed to and from the IPIC; the interrupts are sent to the PowerPC core. The DMA controller can optionally (depending on the programming of the DMA registers) steer its interrupt to the PCI host through the $\overline{\text{PCI\_INTA}}$ signal.

In this mode all machine check interrupts are gathered by the IPIC unit and sent to the PowerPC core. If the device performs as a PCI host, the interrupts of the other PCI agents should be connected to the implementation's $\overline{\text{IRQx}}$ signals and treated like normal external interrupts (sent to the core).

8.3.2 Core Disable Mode

In core disable mode, all internal interrupts (including those from the PCI block) are routed to and from the IPIC, the interrupts are then sent through the $\overline{\text{PCI\_INTA}}$ signal to the PCI host CPU. Note that the core interrupt signal is masked. The user should use in this mode only the $\overline{int}$ output interrupt type (should not

use *cint* or *smi* output interrupt types) to read an updated SIVCR. (See Section 8.5.7, “System Internal Interrupt Control Register (SICNR),” and Section 8.5.12, “System External Interrupt Control Register (SECNR).”)

In this mode, machine check interrupts are driven either on PCI_INTA or on MCP_OUT as level-sensitive interrupts. SERCR[MCPR] (see Section 8.5.15, “System Error Control Register (SERCR)”) controls which external signal is used.

8.4 External Signal Description

The following sections provide an overview and detailed descriptions of the IPIC signals.

8.4.1 Overview

The device has distinct external interrupt request input signals (IRQ[0:7]) and one interrupt request output signal (PCI_INTA). The IPIC interface signals are defined in Table 8-1.

Table 8-1. IPIC Signal Properties

Name	Port	Function	I/O	Reset	Requires Pull Up
IRQ[0:7]	IRQ[0:7]	External interrupts	I	—	Yes
PCI_INTA	PCI_INTA	Interrupt request output	O	Z	Yes
MCP_OUT	MCP_OUT	Interrupt request output	O	Z	Yes

8.4.2 Detailed Signal Descriptions

Table 8-2 provides detailed descriptions of the external IPIC signals.

Table 8-2. IPIC External Signals—Detailed Signal Descriptions

Signal	I/O	Description	
IRQ[0:7]	I	Interrupt request 0–7. The sense (level or edge) of each of these signals is programmable. All of these inputs can be driven completely asynchronously.	
		State Meaning	Asserted—When an external interrupt request signal is asserted the priority is checked by the IPIC unit, and the interrupt is conditionally passed to the processor. Negated—There is no incoming interrupt from that source.
		Timing	Assertion—All of these inputs can be asserted completely asynchronously. Negation—Interrupts programmed as level-sensitive must remain asserted until serviced.

Table 8-2. IPIC External Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_INTA	OD	Interrupt request out. Active-low, open drain. When the IPIC is programmed in core disable mode, this output reflects the raw interrupts generated by on-chip sources. See Section 8.3, "Modes of Operation," for details.	
		State Meaning	Asserted—At least one interrupt is currently being signalled to the external system. Negated—Indicates no interrupt source currently routed to IRQ_OUT.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of IRQ_OUT occur asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 3 system bus clock cycles after the interrupt occurs. External interrupt source: 4 cycles after the interrupt occurs. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 3 system bus clock cycles. External interrupt: 4 cycles.
MCP_OUT	OD	Non-maskable Interrupt (machine check) request out. Active-low, open drain. When the IPIC is programmed in core disable mode, this output reflects the *mcp* interrupts generated by on-chip sources. See Section 8.3, "Modes of Operation."	
		State Meaning	Asserted—At least one machine check interrupt is currently being signalled to the external system. Negated—Indicates no interrupt source currently routed to MCP_OUT.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of MCP_OUT occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 system bus clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 2 system bus clock cycles. External interrupt: 4 cycles.

8.5 Memory Map/Register Definition

The IPIC programmable register map occupies 256 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All IPIC registers are 32 bits wide and they are located on 32-bit address boundaries. Software can perform byte, half-word or word accesses to any IPIC registers. All addresses used in this chapter are offsets from the IPIC base, as defined in Chapter 2, "Memory Map."

Table 8-3 shows the memory map of the IPIC unit. A special set of registers are the QUICC Engine Ports Interrupts registers. These registers have a different base address in the global memory map, and they are detailed in Table 8-4.

Table 8-3. IPIC Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	System global interrupt configuration register (SICFR)	R/W	0x0000_0000	8.5.1/8-8
0x04	System regular interrupt vector register (SIVCR)	R	0x0000_0000	8.5.2/8-9
0x08	System internal interrupt pending register (SIPNR_H)	R	0x0000_0000	8.5.3/8-11
0x0C	System internal interrupt pending register (SIPNR_L)	R	0x0000_0000	8.5.3/8-11

Table 8-3. IPIC Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x10	System internal interrupt group A priority register (SIPRR_A)	R/W	0x0530_9770	8.5.4/8-14
0x1C	System internal interrupt group D priority register (SIPRR_D)	R/W	0x0530_9770	8.5.5/8-14
0x20	System internal interrupt mask register (SIMSR_H)	R/W	0x0000_0000	8.5.6/8-15
0x24	System internal interrupt mask register (SIMSR_L)	R/W	0x0000_0000	8.5.6/8-15
0x28	System internal interrupt control register (SICNR)	R/W	0x0000_0000	8.5.7/8-17
0x2C	System external interrupt pending register (SEPNR)	R/W	Special	8.5.8/8-18
0x30	System mixed interrupt group A priority register (SMPRR_A)	R/W	0x0530_9770	8.5.9/8-18
0x34	System mixed interrupt group B priority register (SMPRR_B)	R/W	0x0530_9770	8.5.10/8-19
0x38	System external interrupt mask register (SEMSR)	R/W	0x0000_0000	8.5.11/8-20
0x3C	System external interrupt control register (SECNR)	R/W	0x0000_0000	8.5.12/8-21
0x40	System error status register (SERSR)	R/W	0x0000_0000	8.5.13/8-22
0x44	System error mask register (SERMR)	R/W	—	8.5.14/8-23
0x48	System error control register (SERCR)	R/W	0x0000_0000	8.5.15/8-24
0x4C–0x4F	Reserved	—	—	—
0x50	System internal interrupt force register (SIFCR_H)	R/W	0x0000_0000	8.5.16/8-25
0x54	System internal interrupt force register (SIFCR_L)	R/W	0x0000_0000	8.5.16/8-25
0x58	System external interrupt force register (SEFCR)	R/W	0x0000_0000	8.5.17/8-26
0x5C	System error force register (SERFR)	R/W	0x0000_0000	8.5.18/8-26
0x60	System critical interrupt vector register (SCVCR)	R	0x0000_0000	8.5.19/8-27
0x64	System management interrupt vector register (SMVCR)	R	0x0000_0000	8.5.20/8-27
0x68–0xFF	Reserved	—	—	—

Table 8-4. QUICC Engine Ports Interrupts Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x0C	QUICC Engine ports interrupt event register (CEPIER)	w1c	Special	8.5.21/8-28
0x10	QUICC Engine ports interrupt mask register (CEPIMR)	R/W	0x0000_0000	8.5.22/8-29
0x14	QUICC Engine ports interrupt control register (CEPICR)	R/W	0x0000_0000	8.5.23/8-30

8.5.1 System Global Interrupt Configuration Register (SICFR)

SICFR, shown in Figure 8-1, defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table. See Table 8-5 for more information.

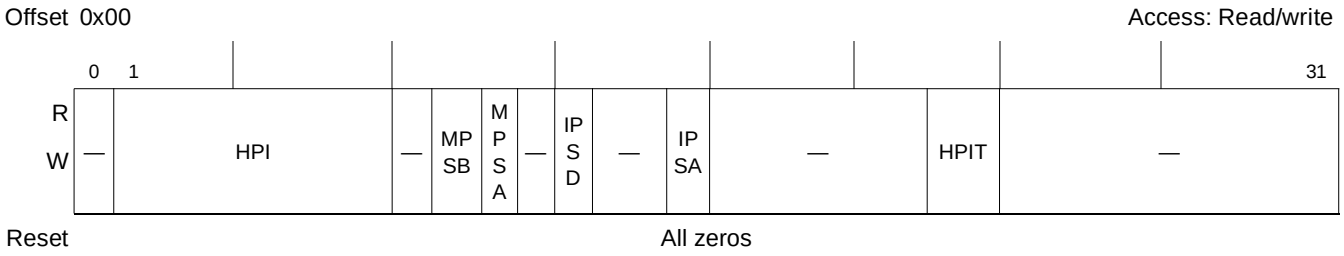


Figure 8-1. System Global Interrupt Configuration Register (SICFR)

Table 8-5 defines the bit fields of SICFR.

Table 8-5. SICFR Field Descriptions

Bits	Name	Description
0	—	Write ignored, read = 0
1–7	HPI	Highest priority interrupt. Specifies the 7-bit unique interrupt number/vector (see Table 8-7) of the single interrupt controller interrupt source that is advanced to the highest priority in the IPIC priority table (see Table 8-35). HPI can be modified dynamically.
8	—	Write ignored, read = 0
9	MPSB	Mixed interrupts priority scheme for group B. Selects the relative MIXB priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXBs are grouped by priority at the top of the table. 1 Spread. The MIXBs are spread by priority in the table.
10	MPSA	Mixed interrupts priority scheme for group A. Selects the relative MIXA priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXAs are grouped by priority at the top of the table. 1 Spread. The MIXAs are spread by priority in the table.
11	—	Write ignored, read = 0
12	IPSD	Internal interrupts priority scheme for group D. Selects the relative SYSD priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSDs are grouped by priority at the top of the table. 1 Spread. The SYSDs are spread by priority in the table.
13–14	—	Write ignored, read = 0
15	IPSA	Internal interrupts priority scheme for group A. Selects the relative SYSA priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSAs are grouped by priority at the top of the table. 1 Spread. The SYSAs are spread by priority in the table.
16–21	—	Write ignored, read = 0

Table 8-5. SICFR Field Descriptions (continued)

Bits	Name	Description
22–23	HPIT	HPI priority position IPIC output interrupt type. Defines which type of IPIC output interrupt signal (*int*, *cint*, or *smi*) asserts its request to the core in the HPI priority position. These bits cannot be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it won't happen during the change). The definition of HPIT is as follows: 00 *int* request is asserted to the core for HPI. 01 *smi* request is asserted to the core for HPI. 10 *cint* request is asserted to the core for HPI. 11 Reserved.
24–31	—	Write ignored, read = 0

8.5.2 System Global Interrupt Vector Register (SIVCR)

SIVCR, shown in Figure 8-2, contains a 7-bit code (Table 8-6) representing the regular unmasked interrupt source (INT) of the highest priority level.

NOTE

Note that in core disabled mode the user should use SIVCR only in order to read an updated interrupt vector (SCVCR and SMVCR should not be used).

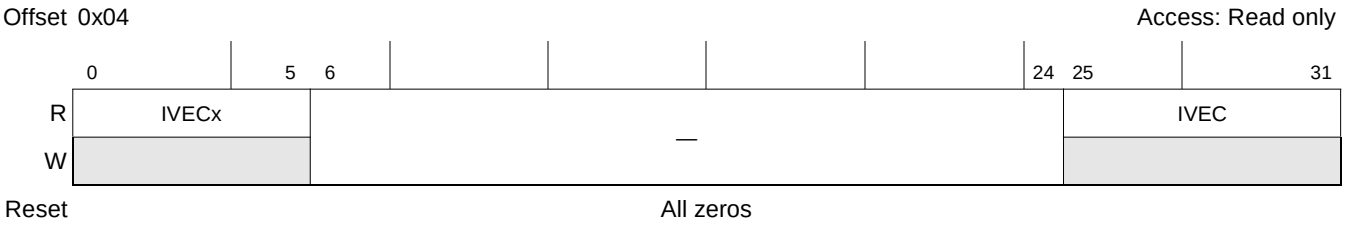


Figure 8-2. System Global Interrupt Vector Register (SIVCR)

Table 8-6 defines the bit fields of SIVCR.

Table 8-6. SIVCR Field Descriptions

Bits	Name	Description
0–5	IVECx	Backward (MPC8260) compatible regular interrupt vector. Specifies a 6-bit unique number of the IPIC's highest priority regular interrupt source, pending to the core. When a regular interrupt request occurs, SIVCR can be read. If there are multiple regular interrupt sources, SIVCR latches the highest priority regular interrupt. Note that IVECx field correctly reflects only the first 64 interrupt vectors (See Table 8-7 for details). The value of SIVEC cannot change while it is being read.
6–24	—	Write ignored, read = 0
25–31	IVEC	Regular interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority regular interrupt source, pending to the core. Note that the when a regular interrupt request occurs, SIVCR can be read. If there are multiple regular interrupt sources, SIVCR latches the highest priority regular interrupt. Note that the IVEC field correctly reflects all interrupt vectors (see Table 8-7 for details). The value of SIVCR cannot change while it is being read.

Table 8-7 shows the definition of IVEC.

Table 8-7. IVEC/CVEC/MVEC Field Definition

Interrupt ID Number	Interrupt Meaning	Interrupt Vector
0	Error (no interrupt)	0b000_0000
1–8	Reserved	0b000_0001–0b000_1000
9	UART1	0b000_1001
10	UART2	0b000_1010
11	SEC	0b000_1011
12–13	Reserved	0b000_1100–0b000_1101
14	I2C	0b000_1110
15	Reserved	0b000_1111
16	Reserved	0b001_0000
17	IRQ1	0b001_0001
18	IRQ2	0b001_0010
19	IRQ3	0b001_0011
20	IRQ4	0b001_0100
21	IRQ5	0b001_0101
22	IRQ6	0b001_0110
23	IRQ7	0b001_0111
24–31	Reserved	0b001_1000–0b001_1111
32	QUICC Engine High	0b010_0000
33	QUICC Engine Low	0b010_0001
34–47	Reserved	0b010_0010–0b010_1111
48	IRQ0	0b011_0000
49–63	Reserved	0b011_0001–0b011_1111
64	RTC SEC	0b100_0000
65	PIT	0b100_0001
66	PCI	0b100_0010
67	Reserved	0b100_0011
68	RTC ALR	0b100_0100
69	MU	0b100_0101
70	SBA	0b100_0110
71	DMA	0b100_0111
72	GTM4	0b100_1000
73	GTM8	0b100_1001
74	QUICC Engine Ports	0b100_1010
75	Reserved	0b100_1011

Table 8-7. IVEC/CVEC/MVEC Field Definition (continued)

Interrupt ID Number	Interrupt Meaning	Interrupt Vector
76	DDR	0b100_1100
77	LBC	0b100_1101
78	GTM2	0b100_1110
79	GTM6	0b100_1111
80	PMC	0b101_0000
81–83	Reserved	0b101_0001–0b101_0011
84	GTM3	0b101_0100
85	GTM7	0b101_0101
86–89	Reserved	0b101_0110–0b101_1001
90	GTM1	0b101_1010
91	GTM5	0b101_1011
92–127	Reserved	0b101_1100–0b111_1111

8.5.3 System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L)

Each bit in SIPNR_H and SIPNR_L, shown in Figure 8-3 and Figure 8-4, may be assigned an internal interrupt source. (Implemented bits are listed in Table 8-8.) When an interrupt request is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit.

Note that SIPNR bit positions are not changed according to relative priority.

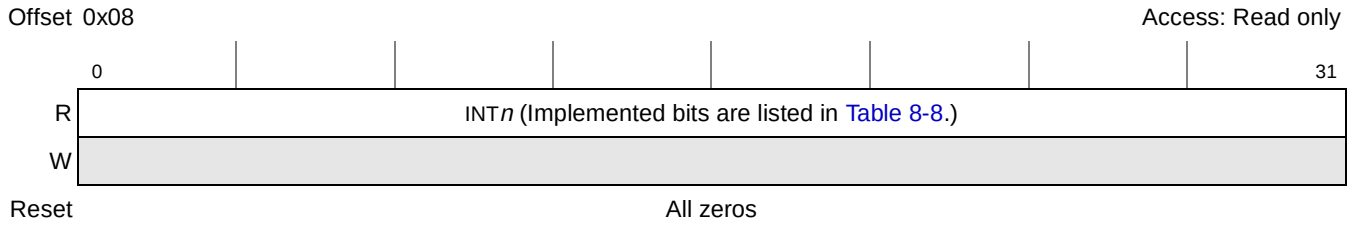


Figure 8-3. System Internal Interrupt Pending Register (SIPNR_H)

Table 8-8 lists implemented SIPNR_H fields. Note that these field descriptions are also valid for SIFCR_H and SIMSR_H.

Table 8-8. SIPNR_H/SIFCR_H/SIMSR_H Bit Assignments

Bits	Field
0	QE High
1	QE Low
2	—
3	—

Table 8-8. SIPNR_H/SIFCR_H/SIMSR_H Bit Assignments (continued)

Bits	Field
4	—
5	—
6	—
7	—
8–23	—
24	UART1
25	UART2
26	SEC
27–28	—
29	I2C
30	—
31	—

Table 8-9 defines the bit fields of SIPNR_H.

Table 8-9. SIPNR_H Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit (listed in Table 8-8) corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit. SIPNR bits are read only. Writing to this register has no effect. Note that the SIPNR bit positions are not changed according to their relative priority. For unimplemented bits, writes are ignored, read = 0.

SIPNR_L is shown in Figure 8-3.

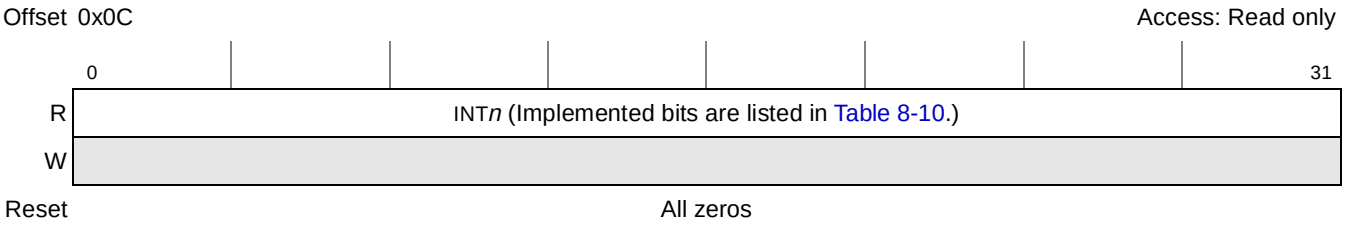


Figure 8-4. System Internal Interrupt Pending Register (SIPNR_L)

Table 8-10 lists implemented SIPNR_L fields. Note that these field assignments are also valid for SIFCR_L and SIMSR_L.

Table 8-10. SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments

Bits	Field
0	RTC SEC
1	PIT
2	PCI
3	—

Table 8-10. SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments (continued)

Bits	Field
4	RTC ALR
5	MU
6	SBA
7	DMA
8	GTM4
9	GTM8
10	QE Ports
11	—
12	DDR
13	LBC
14	GTM2
15	GTM6
16	PMC
17–19	—
20	GTM3
21	GTM7
22–25	—
26	GTM1
27	GTM5
28–30	—
31	—

Table 8-11 defines the bit fields of SIPNR_L.

Table 8-11. SIPNR_L Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit (listed in Table 8-10) corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit. SIPNR bits are read only. Writing to this register has no effect. Note that the SIPNR bit positions are not changed according to their relative priority. For unimplemented bits, writes are ignored, read = 0.

8.5.4 System Internal Interrupt Group A Priority Register (SIPRR_A)

The system internal interrupt group A priority register (SIPRR_A), shown in Figure 8-5, defines the priority between QE High and QE Low internal interrupt signals.

Offset 0x10 Access: Read/write

	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R W	SYSA0P	SYSA1P	SYSA2P	SYSA3P	—	SYSA4P	SYSA5P	SYSA6P	SYSA7P	—
Reset	0 0 0	0	1 0 1	0 0 1 1	0 0 0 0	1 0 0 1	0 1 1 1	0	1 1 1	0 0 0 0

Figure 8-5. System Internal Interrupt Group A Priority Register (SIPRR_A)

Table 8-12 defines the bit fields of SIPRR_A.

Table 8-12. SIPRR_A Field Descriptions

Bits	Name	Description
0–2	SYSA0P	SYSA0 priority order. Defines which interrupt source asserts its request in the SYSA0 priority position. The user should not program the same code to multiple priority positions (0–7). These bits can be changed dynamically. The definition of SYSA0P is as follows: 000 QE High asserts its request in the SYSA0 position. 001 QE Low asserts its request in the SYSA0 position. 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
3–11, 16–27	SYSA1P–SYSA7P	Same as SYSA0P, but for SYSA1P–SYSA7P.
12–15, 28–31	—	Write ignored, read = 0

8.5.5 System Internal Interrupt Group D Priority Register (SIPRR_D)

SIPRR_D, shown in Figure 8-6, defines the priority among the interrupt sources listed in Table 8-13.

Offset 0x1C Access: Read/write

	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R W	SYSD0P	SYSD1P	SYSD2P	SYSD3P	—	SYSD4P	SYSD5P	SYSD6P	SYSD7P	—
Reset	0 0 0	0	1 0 1	0 0 1 1	0 0 0 0	1 0 0 1	0 1 1 1	0	1 1 1	0 0 0 0

Figure 8-6. System Internal Interrupt Group D Priority Register (SIPRR_D)

Table 8-13 defines the bit fields of SIPRR_D.

Table 8-13. SIPRR_D Field Descriptions

Bits	Name	Description
0–2	SYSD0P	SYSD0 priority order. Defines which interrupt source asserts its request in the SYSD0 priority position. The user should not program the same code to more than one priority position (0–7). These bits can be changed dynamically. SYSD0P is defined as follows: 000 UART1 asserts its request in the SYSD0 position. 001 UART2 asserts its request in the SYSD0 position. 010 SEC asserts its request in the SYSD0 position. 011 Reserved 100 Reserved 101 I2C asserts its request in the SYSD0 position. 110 Reserved 111 Reserved
3–11, 16–27	SYSD1P–SYSD7P	Same as SYSD0P, but for SYSD1P–SYSD7P.
12–15, 28–31	—	Write ignored, read = 0

8.5.6 System Internal Interrupt Mask Register (SIMSR_H and SIMSR_L)

Each implemented bit in SIMSR_H and SIMSR_L, shown in Figure 8-7 and Figure 8-8, corresponds to an internal interrupt source. The user masks an interrupt by clearing the corresponding SIMSR bit. When an interrupt request occurs, the corresponding SIPNR bit is set, regardless of the SIMSR bit. However, if the corresponding SIMSR bit is cleared, no interrupt request is passed to the core.

When an SIMSR bit is cleared by the user at the same time corresponding interrupt source requests an interrupt service, the request stops. If the user sets the SIMSR bit later, the core processes any pending corresponding interrupt requests according to its priority.

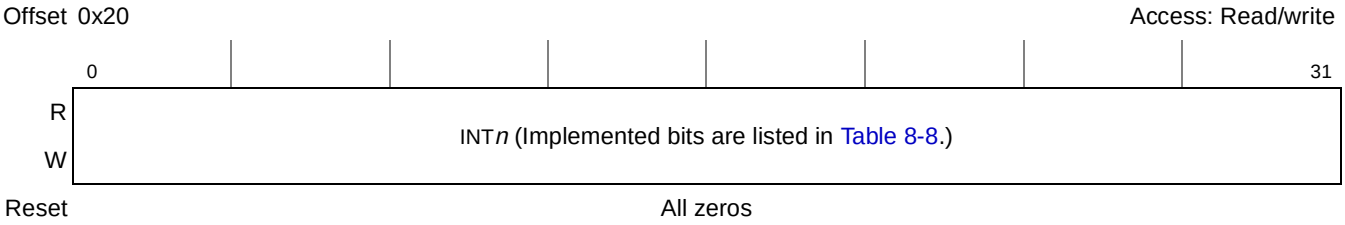


Figure 8-7. System Internal Interrupt Mask Register (SIMSR_H)

Table 8-14 defines the bit fields of SIMSR_H.

Table 8-14. SIMSR_H Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit (listed in Table 8-8) corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SIMSR bit. An interrupt is unmasked (enable) by setting the corresponding SIMSR bit. The SIMSR can be read by the user at any time. **Note:** • SIMSR bit positions do not change according to their relative priority. • The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register. • If an SIMSR bit is masked at the same time that the corresponding SIPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts are pending). Thus, the user should always include an error vector routine, even if it contains only an **rfi** instruction. The error vector cannot be masked. Unimplemented bits, shown as reserved in Table 8-8, are ignored on writes; read = 0.

Figure 8-8 shows SIMSR_L.

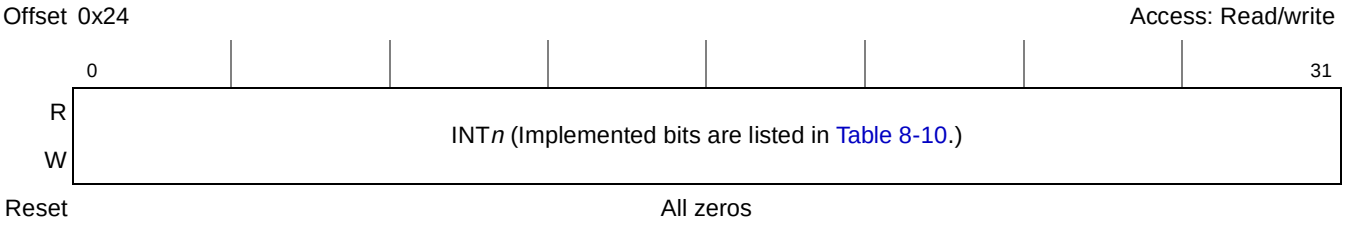


Figure 8-8. System Internal Interrupt Mask Register (SIMSR_L)

Table 8-15 defines the bit fields of SIMSR_L.

Table 8-15. SIMSR_L Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit (listed in Table 8-10) corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SIMSR bit. An interrupt is unmasked (enabled) by setting the corresponding SIMSR bit. The SIMSR can be read by the user at any time. **Note:** • SIMSR bit positions are not changed according to their relative priority. • The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register. • If an SIMSR bit is masked at the same time that the corresponding SIPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts are pending). Thus, the user should always include an error Unimplemented bits, shown as reserved in Figure 8-8, are ignored on writes; read = 0.

8.5.7 System Internal Interrupt Control Register (SICNR)

SICNR, shown in Figure 8-9, defines the IPIC output interrupt type ($\overline{\text{int}}$, $\overline{\text{cint}}$, or $\overline{\text{smi}}$) in the SYSA0–SYSA1 and SYSD0–SYSD1 priority positions. All other priority positions assert $\overline{\text{int}}$ to the core.

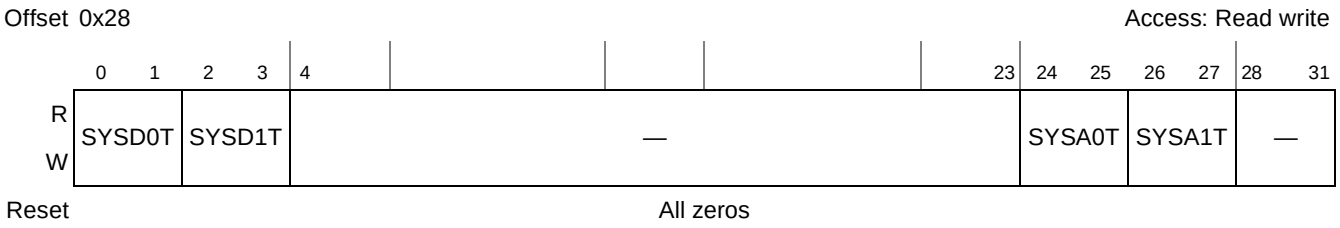


Figure 8-9. System Internal Interrupt Control Register (SICNR)

Table 8-16 defines the bit fields of SICNR.

Table 8-16. SICNR Field Descriptions

Bits	Name	Description
0–1	SYSD0T	SYSD0 priority position IPIC output interrupt type.Defines which type of the IPIC output interrupt signal ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the SYSD0 priority position. These bits cannot be changed dynamically. (to change it, software must make sure the corresponding interrupt source is masked or it cannot happen during the change). The definition of SYSD0T is as follows: 00 $\overline{int}$ request is asserted to the core for SYSD0. 01 $\overline{smi}$ request is asserted to the core for SYSD0. 10 $\overline{cint}$ request is asserted to the core for SYSD0. 11 Reserved
2–3	SYSD1T	Same as SYSD0T, but for SYSD1T.
4–23	—	Write ignored, read = 0
24–25	SYSA0T	SYSA0 priority position IPIC output interrupt type. Defines which type of the IPIC output interrupt signal ($\overline{int}$, $\overline{smi}$, or $\overline{cint}$) asserts its request to the core in the SYSA0 priority position. These bits can not be changed dynamically. (If s/w really wants to change it, it has to make sure the corresponding interrupt source is masked or it won't happen during the change). The definition of SYSA0T is as follows: 00 $\overline{int}$ request is asserted to the core for SYSA0. 01 $\overline{smi}$ request is asserted to the core for SYSA0. 10 $\overline{cint}$ request is asserted to the core for SYSA0. 11 Reserved.
26–27	SYSA1T	Same as SYSA0T, but for SYSA1T
28–31	—	Write ignored, read = 0

8.5.8 System External Interrupt Pending Register (SEPNR)

Each bit in the SEPNR, shown in Figure 8-10, corresponds to an external interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SEPNR bit.

Offset 0x2C Access: Read/write

	0	1	2	3	4	5	6	7	8–15
R	IRQ0[1]	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5	IRQ6	IRQ7	—
W									
Reset	The reset values of implemented bits reflect the values of the external IRQ signals. Reserved bits are zeros. [2]								

	16–31
R	—
W	
Reset	All zeros

1 This bit is valid only if the IRQ0 signal is configured as an external maskable interrupt (SEMSR[SIRQ0] = 0)

2 The user should drive all IRQ inputs to an inactive state prior to reset negation

Figure 8-10. System External Interrupt Pending Register (SEPNR)

Table 8-17 defines the bit fields of SEPNR.

Table 8-17. SEPNR Field Descriptions

Bits	Name	Description
0–7	IRQ*n*	Each bit corresponds to an external interrupt source. When an external interrupt is received, the interrupt controller sets the corresponding SEPNR bit. When a pending interrupt is handled, the user must clear the corresponding SEPNR bit. For level triggered cases, the software needs to cause the $\overline{\text{IRQn}}$ to negate which automatically clears the bit in SEPNR. For edge-triggered cases, the software needs to clear the corresponding bit in SEPNR. SEPNR 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 SEPNR bit positions are not changed according to their relative priority.
8–31	—	Write ignored, read = 0

8.5.9 System Mixed Interrupt Group A Priority Register (SMPRR_A)

The SMPRR_A, shown in Figure 8-11, defines the priority among the sources listed in Table 8-18.

Offset 0x30 Access: Read/write

	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R	MIXA0P	MIXA1P	MIXA2P	MIXA3P	—	MIXA4P	MIXA5P	MIXA6P	MIXA7P	—
W										
Reset	0 0 0	0 1 0	1 0 0	1 1 0	0 0 0 0	1 0 0	1 0 1	1 1 0	1 1 1	0 0 0 0

Figure 8-11. System Mixed Interrupt Group A Priority Register (SMPRR_A)

Table 8-18 defines the bit fields of SMPRR_A.

Table 8-18. SMPRR_A Field Descriptions

Bits	Name	Description
0–2	MIXA0P	MIXA0 priority order. Defines which interrupt source asserts its request in the MIXA0 priority position. The user must not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXA0P is as follows: 000 RTC SEC asserts its request to the MIXA0 position. 001 PIT asserts its request to the MIXA0 position. 010 PCI asserts its request to the MIXA0 position. 011 Reserved. 100 IRQ0 asserts its request to the MIXA0 position. This field for MIXA0 position is valid (must not be ignored) if IRQ0 signal configured as an external maskable interrupt (SEMSR[SIRQ0] = 0). 101 IRQ1 asserts its request to the MIXA0 position. 110 IRQ2 asserts its request to the MIXA0 position. 111 IRQ3 asserts its request to the MIXA0 position.
3–11, 16–27	MIXA1P–MIXA7P	Same as MIXA0P, but for MIXA1P–MIXA7P.
12–15, 28–31	—	Write ignored, read = 0

8.5.10 System Mixed Interrupt Group B Priority Register (SMPRR_B)

SMPRR_B, shown in Figure 8-12, defines the priority among the sources listed in Table 8-19.

Offset 0x34 Access: Read/write

	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R W	MIXB0P	MIXB1P	MIXB2P	MIXB3P	—	MIXB4P	MIXB5P	MIXB6P	MIXB7P	—
Reset	0 0 0 0	0 1 0 1	0 0 1 1	0 0 0 0	1 0 0 1	0 1 1 1	0 1 1 1	0 0 0 0		

Figure 8-12. System Mixed Interrupt Group B Priority Register (SMPRR_B)

Table 8-19 defines the bit fields of SMPRR_B.

Table 8-19. SMPRR_B Field Descriptions

Bits	Name	Description
0–2 3–11, 16–27	MIXB*n*P	MIXB*n* priority order. Defines which interrupt source asserts its request in the MIXB*n* priority position. The user must not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXB*n*P is as follows: 000 RTC ALR asserts its request to the MIXB*n* position. 001 MU asserts its request to the MIXB*n* position. 010 SBA asserts its request to the MIXB*n* position. 011 DMA asserts its request to the MIXB*n* position. 100 IRQ4 asserts its request to the MIXB*n* position. 101 IRQ5 asserts its request to the MIXB*n* position. 110 IRQ6 asserts its request to the MIXB*n* position. 111 IRQ7 asserts its request to the MIXB*n* position.
12–15, 28–31	—	Write ignored, read = 0

8.5.11 System External Interrupt Mask Register (SEMSR)

Each bit in the system external interrupt mask register (SEMSR), shown in Figure 8-10, corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SEMSR bit. An interrupt is unmasked (enabled) by setting the corresponding SEMSR bit.

When an external interrupt request occurs, the corresponding SEPNR bit is set regardless of the setting of the corresponding SEMSR bit. However, if the corresponding SEMSR bit is cleared, no interrupt request is passed to the core.

When an SEMSR bit is cleared by the user at the same time that an interrupt source requests an interrupt service, the request stops. If the user sets the SEMSR bit later, a previously pending interrupt request is processed by the core according to its assigned priority. SEMSR can be read by the user at any time.

Offset 0x38 Access: Read/write

	0	1	2	3	4	5			15
R W	IRQ0[1]	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5	IRQ6	IRQ7	—
Reset	The reset values of implemented bits reflect the values of the external IRQ signals. Reserved bits are zeros. [2]								

	16	17			31
R W	SIRQ0	—			
Reset	All zeros				

1 This bit is valid only if the IRQ0 signal is configured as an external maskable interrupt (SEMSR[SIRQ0] = 0)

2 The user should drive all IRQ inputs to an inactive state prior to reset negation

Figure 8-13. System External Interrupt Mask Register (SEMSR)

Table 8-20 defines the bit fields of SEMSR.

Table 8-20. SEMSR Field Descriptions

Bits	Name	Description
0–7	—	Each bit corresponds to an external interrupt source. The user masks an interrupt by clearing the SEMSR bit. An interrupt can be enabled by setting the corresponding SEMSR bit. SEMSR can be read by the user at any time. **Note:** • SEMSR bit positions are not affected by their relative priority. • The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register. • If an SEMSR bit is masked at the same time that the corresponding SEPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Thus, the user must always include an error vector routine, even if it contains only an **rfi** instruction. The error vector cannot be masked.
8–15	—	Write ignored, read = 0
16	SIRQ0	Steer IRQ0. 0 IRQ0 is used as external interrupt request 1 IRQ0 is used as external MCP request
17–31	—	Write ignored, read = 0

8.5.12 System External Interrupt Control Register (SECNR)

SECNR, shown in Figure 8-14, defines the edge detect mode for external $\overline{\text{IRQ}}n$ interrupt signals and determines whether the corresponding $\overline{\text{IRQ}}n$ signal asserts an interrupt request upon either a high-to-low change or assertion on the pin. It also defines the IPIC output interrupt type ($\overline{\text{int}}$, $\overline{\text{cint}}$, or $\overline{\text{smi}}$) in the MIXA0–MIXA1 and MIXB0–MIXB1 priority positions.

Note that in core disabled mode of operation the user should use the $\overline{int}$ output interrupt type (should not use $\overline{cint}$ or $\overline{smi}$ output interrupt types) in order to read an updated SIVCR.

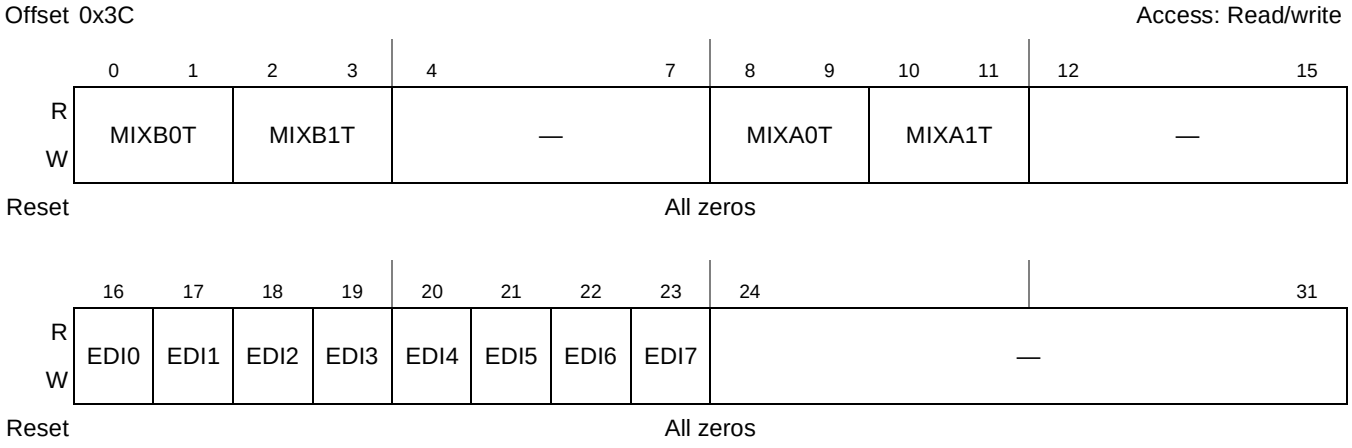


Figure 8-14. System External Interrupt Control Register (SECNR)

Table 8-21 defines the bit fields of SECNR.

Table 8-21. SECNR Field Descriptions

Bits	Name	Description
0–1	MIXB0T	MIXB0 priority position IPIC output interrupt type. Defines which type of the IPIC output interrupt signal ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the MIXB0 priority position. These bits can be changed dynamically. The definition of MIXB0T is as follows: 00 $\overline{int}$ request is asserted to the core for MIXB0. 01 $\overline{smi}$ request is asserted to the core for MIXB0. 10 $\overline{cint}$ request is asserted to the core for MIXB0. 11 Reserved
2–3	MIXB1T	Same as MIXB0T, but for MIXB1T.
4–7	—	Write ignored, read = 0
8–9	MIXA0T	MIXA0 priority position IPIC output interrupt Type. Defines which type of the IPIC output interrupt signal ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the MIXA0 priority position. These bits can be changed dynamically. The definition of MIXA0T is as follows: 00 $\overline{int}$ request is asserted to the core for MIXA0. 01 $\overline{smi}$ request is asserted to the core for MIXA0. 10 $\overline{cint}$ request is asserted to the core for MIXA0. 11 Reserved
10–11	MIXA1T	Same as MIXA0T, but for MIXA1T.
12–15	—	Write ignored, read = 0
16–23	EDIx	Each bit defines the edge detect mode for the external $\overline{IRQ}n$ interrupt signals, determines whether the corresponding $\overline{IRQ}n$ signal asserts an interrupt request upon either a high-to-low change or low assertion on the pin. The corresponding $\overline{IRQ}n$ signal asserts an interrupt request as follows: 0 Low assertion on $\overline{IRQ}n$ generates an interrupt request (level sensitive). 1 High-to-low change on $\overline{IRQ}n$ generates an interrupt request (edge sensitive).
24–31	—	Write ignored, read = 0

8.5.13 System Error Status Register (SERSR)

The bits in the SERSR, shown in Figure 8-15, correspond to the external and internal non-maskable error source machine check (mcp) conditions listed in Table 8-22. When an error interrupt signal is received, the interrupt controller sets the corresponding SERSR bit.

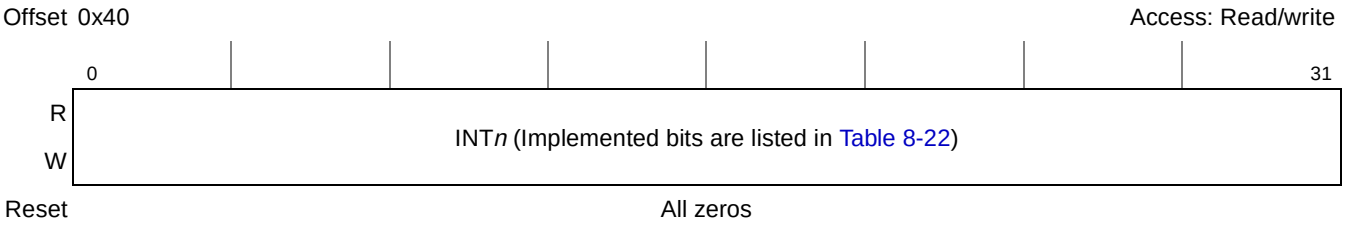


Figure 8-15. System Error Status Register (SERSR)

Table 8-22 lists the implemented SERSR bits. Note that these field assignments are valid for SERMR and SERFR.

Table 8-22. SERSR/SERMR/SERFR Bit Assignments

Bits	Field
0	IRQ0[1]
1	WDT
2	SBA
3	CIEE
4	CMEE
5	PCI
6	—
7	MU
8–14	—
15	—
16–31	—

1 This bit is valid only if the IRQ0 signal is configured as an external MCP interrupt (SEMSR[SIRQ0] = 1)

Table 8-23 defines the bit fields of SERSR.

Table 8-23. SERSR Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit in the SERSR, listed in Table 8-22, corresponds to an external and an internal error source (mcp). When an error interrupt signal is received, the interrupt controller sets the corresponding SERSR bit. SERSR bits are cleared by writing ones to them. Unmasked event register bits should be cleared before clearing SERSR bits. Because the user can only clear bits in this register, writing zeros to this register has no effect. SERSR bits are cleared by power-on reset. Subsequent soft and hard resets do not affect SERSR bit states. For unimplemented bits (listed as reserved in Table 8-22), writes are ignored, read = 0

8.5.14 System Error Mask Register (SERMR)

Each implemented bit in SERMR, shown in Figure 8-16, corresponds to an external and an internal $\overline{mcp}$ source (MCP). The user masks an MCP by clearing and enables an interrupt by setting the corresponding SERMR bit. When a masked MCP occurs, the corresponding SERSR bit is set, regardless of the setting of the corresponding SERMR bit although no MCP request is passed to the core in this case. The SERMR can be read by the user at any time.

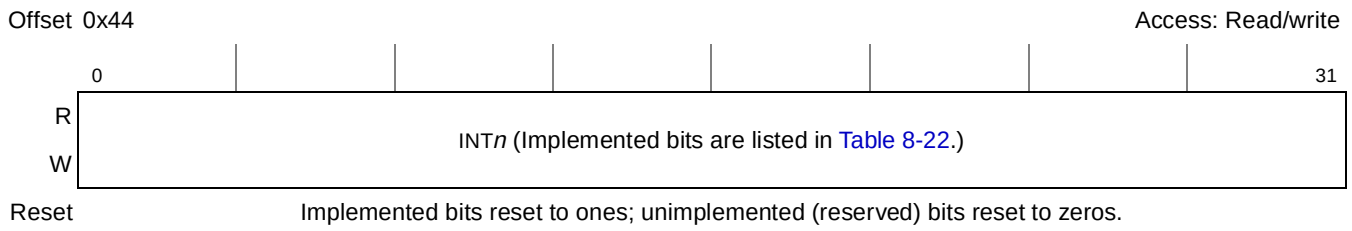


Figure 8-16. System Error Mask Register (SERMR)

Table 8-24 defines the bit fields of SERMR.

Table 8-24. SERMR Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented SERMR bit, listed in Table 8-22, corresponds to an external and an internal MCP source. The user masks an MCP by clearing and enables an interrupt by setting the corresponding SERMR bit. When a masked MCP occurs, the corresponding SERSR bit is set, regardless of the setting of the SERMR bit although no MCP request is passed to the core. The SERMR can be read by the user at any time. Writes to unimplemented (reserved) bits are ignored; read = 0

8.5.15 System Error Control Register (SERCR)

SERCR, shown in Figure 8-17, defines the control bits that route MCP requests in core disable mode to either $\overline{\text{MCP\_OUT}}$ or $\overline{\text{PCI\_INTA}}$ in core-disable mode.

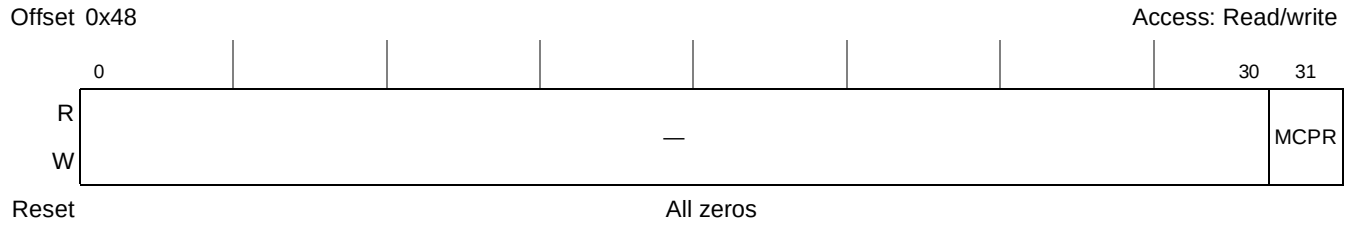


Figure 8-17. System Error Control Register (SERCR)

Table 8-25 defines the bit fields of SERCR.

Table 8-25. SERCR Field Descriptions

Bits	Name	Description
0–30	—	Write ignored, read = 0
31	MCPR	MCP route. Route MCP request to either $\overline{\text{MCP\_OUT}}$ or $\overline{\text{PCI\_INTA}}$ (in core disable mode). 0 MCP routed to $\overline{\text{PCI\_INTA}}$ (in core disable mode). 1 MCP routed to $\overline{\text{MCP\_OUT}}$ (in core disable mode).

8.5.16 System Internal Interrupt Force Registers (SIFCR_H and SIFCR_L)

Each implemented bit SIFCR_H and SIFCR_L, shown in Figure 8-18 and Figure 8-19, corresponds to an internal interrupt source. When a bit is set, the interrupt controller generates the corresponding interrupt (sets the corresponding SIPNR bit).The SIFCR can be read by the user at any time.

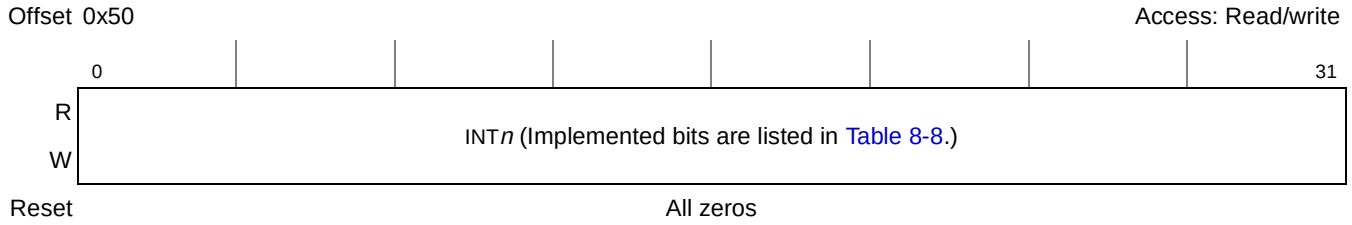


Figure 8-18. System Internal Interrupt Force Register (SIFCR_H)

Table 8-26 defines the bit fields of SIFCR_H.

Table 8-26. SIFCR_H Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit, listed in Table 8-8, corresponds to an internal interrupt source. The user forces an interrupt by setting the corresponding SIFCR*x* bit. SIFCR*n* bit positions are not changed according to their relative priority. Writes to unimplemented (reserved) bits are ignored; read = 0

SIFCR_L is shown in Figure 8-19.

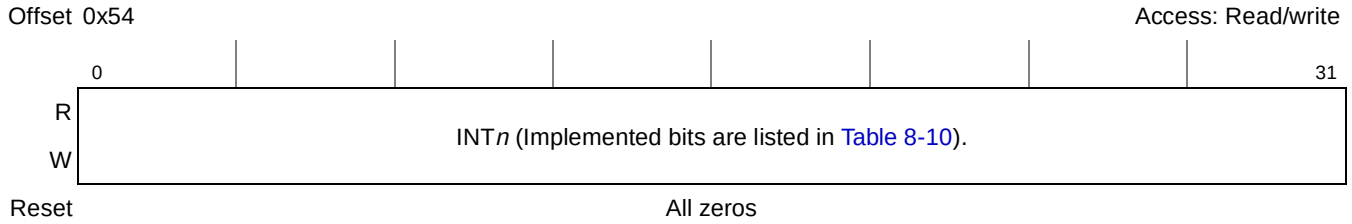


Figure 8-19. System Internal Interrupt Force Register (SIFCR_L)

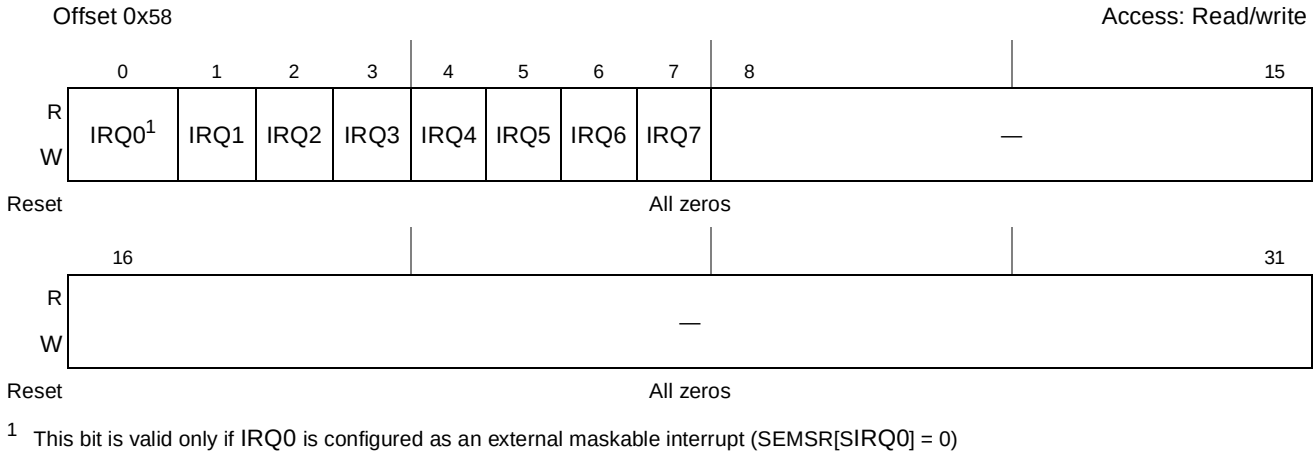
Table 8-27 defines the bit fields of SIFCR_L.

Table 8-27. SIFCR_L Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit, listed in Table 8-10, corresponds to an internal interrupt source. The user forces an interrupt by setting the corresponding SIFCR*x* bit. SIFCR*x* bit positions are not changed according to their relative priority. Writes to unimplemented (reserved) bits are ignored; read = 0

8.5.17 System External Interrupt Force Register (SEFCR)

Each implemented bit in SEFCR, shown in Figure 8-20, corresponds to an external interrupt source. When a bit is set, the interrupt controller generates the corresponding external interrupt (sets the corresponding SEPNR bit). SEFCR can be read by the user at any time.



[1] This bit is valid only if IRQ0 is configured as an external maskable interrupt (SEMSR[SIRQ0] = 0)

Figure 8-20. System External Interrupt Force Register (SEFCR)

Table 8-28 defines the bit fields of SEFCR.

Table 8-28. SEFCR Field Descriptions

Bits	Name	Description
0–7	IRQ*n*	Each bit corresponds to an external interrupt source. The user force an interrupt by setting the SEFCR bit. **Note:** SEFCR bit positions are not affected by their relative priority.
8–31	—	Write ignored, read = 0

8.5.18 System Error Force Register (SERFR)

Each bit in the system error force register (SERFR), shown in Figure 8-21, corresponds to an external MCP source. When a bit is set, the interrupt controller generates the corresponding MCP interrupt (sets the corresponding SERSR bit). The SERFR can be read by the user at any time.

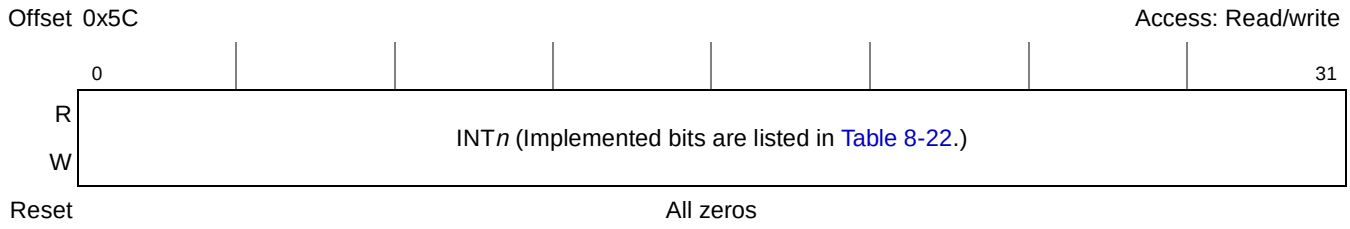


Figure 8-21. System Error Status Register (SERFR)

Table 8-29 defines the bit fields of SERFR.

Table 8-29. SERFR Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit, listed in Table 8-22, corresponds to an external MCP source. The user forces an MCP by setting the SERFR bit. SERFR bit positions are not affected by their relative priority. Attempts to write to unimplemented (reserved) bits are ignored; read = 0

8.5.19 System Critical Interrupt Vector Register (SCVCR)

SCVCR, shown in Figure 8-22, contains a 7-bit code (Table 8-30) representing the unmasked critical interrupt ($\overline{\text{CINT}}$) source of the highest priority level.

Note that in core-disabled mode the user should use SIVCR only to read an updated interrupt vector (SCVCR should not be used).

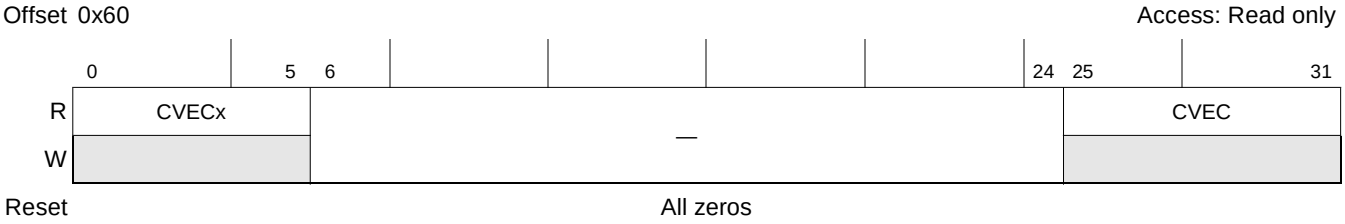


Figure 8-22. System Critical Interrupt Vector Register (SCVCR)

Table 8-30 defines SCVCR bit fields.

Table 8-30. SCVCR Field Descriptions

Bits	Name	Description
0–5	CVECx	Backward (MPC8260) compatible critical interrupt vector. Specifies a 6-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, SCVCR can be read. If there are multiple critical interrupt sources, SCVCR latches the highest priority critical interrupt. Note that CVECx field will correctly reflect only first 64 interrupt vectors (See Table 8-7 for details). The value of SCVEC cannot change while it is being read.
6–24	—	Write ignored, read = 0
25–31	CVEC	Critical interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, SCVCR can be read. If there are multiple critical interrupt sources, SCVCR latches the highest priority critical interrupt. Note that CVEC field correctly reflects all of the interrupt vectors (See Table 8-7 for details). The value of SCVEC cannot change while it is being read.

8.5.20 System Management Interrupt Vector Register (SMVCR)

SMVCR, shown in Figure 8-23, contains a 7-bit code (Table 8-31) representing the unmasked system management interrupt ($\overline{\text{SMI}}$) source of the highest priority level.

Note that in core disabled mode the user should use SIVCR only read an updated interrupt vector (SMVCR should not be used).

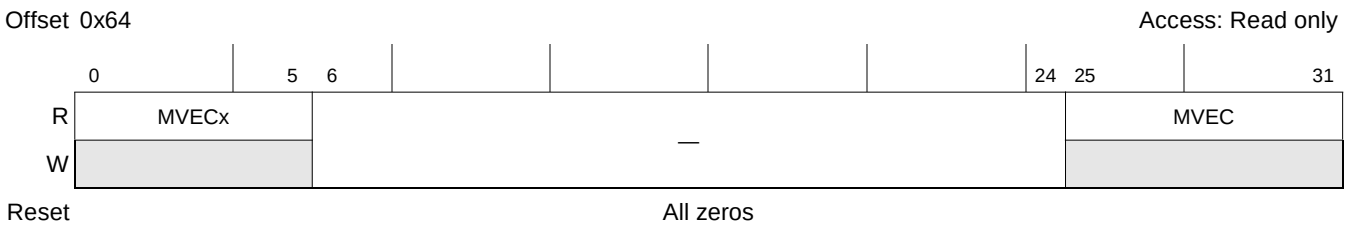


Figure 8-23. System Management Interrupt Vector Register (SMVCR)

Table 8-31 defines the bit fields of SMVCR.

Table 8-31. SMVCR Field Descriptions

Bits	Name	Description
0–5	MVECx	Backward (MPC8260) compatible system management interrupt vector. Specifies a 6-bit unique number of the IPIC's highest priority system management interrupt source, pending to the core. When a system management interrupt request occurs, SMVCR can be read. If there are multiple system management interrupt sources, SMVCR latches the highest priority system management interrupt. Note that MVECx f correctly reflects only the first 64 interrupt vectors (See Table 8-7 for details). The value of SMVEC cannot change while it is being read.
6–24	—	Write ignored, read = 0
25–31	MVEC	System management interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority system management interrupt source, pending to the core. When a system management interrupt request occurs, SMVCR can be read. If there are multiple system management interrupt sources, SMVCR latches the highest priority system management interrupt. Note that MVEC field will correctly reflect all interrupt vectors (See Table 8-7 for details). The value of SMVEC cannot change while it is being read.

8.5.21 QUICC Engine Ports Interrupt Event Register (CEPIER)

The QUICC Engine ports interrupt event register (CEPIER), shown in Figure 8-24, carries information on the QUICC Engine ports events that caused an interrupt. Each bit in CEPIER, corresponds to one QUICC Engine port, which may be the source of an interrupt. CEPIER bits are cleared by writing ones. However, writing zero has no effect.

Offset 0x0C[1] Access: w1c

	0	1	2	3	4	5	6	7	8	9	10–15
R	PA[8]	PA[10]	PA[26]	PA[28]	PB[7]	PB[9]	PB[25]	PB[27]	PC[24]	PC[29]	—
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	Undefined (the user should write 1s to clear before using)										

	16–31
R	—
W	
Reset	Undefined (the user should write 1s to clear before using)

Figure 8-24. QUICC Engine Ports Interrupt Event Register (CEPIER)

[1] Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See Chapter 2, “Memory Map.”

Table 8-32 defines the bit fields of CEPIER.

Table 8-32. CEPIER Bit Settings

Bits	Name	Description
0–9	Px[*n*]	QUICC Engine ports interrupt events. Indicates whether interrupt event occurred on the corresponding QUICC Engine port signal. 0 No interrupt event occurred on the corresponding QUICC Engine port signal. 1 Interrupt event occurred on the corresponding QUICC Engine port signal.
10–31	—	Reserved. Should be cleared.

8.5.22 QUICC Engine Ports Interrupt Mask Register (CEPIMR)

The QUICC Engine ports interrupt mask register (CEPIMR), shown in Figure 8-25, defines the interrupt masking for the individual QUICC Engine ports lines. When a masked interrupt request occurs, the corresponding CEPIER bit is set, regardless of the CEPIMR state. When one or more non-masked interrupt events occur, the ‘QE Ports’ internal event is generated. The ‘QE Ports’ internal event is one source in the SIPNR register (See Section 8.5.3, “System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L).”

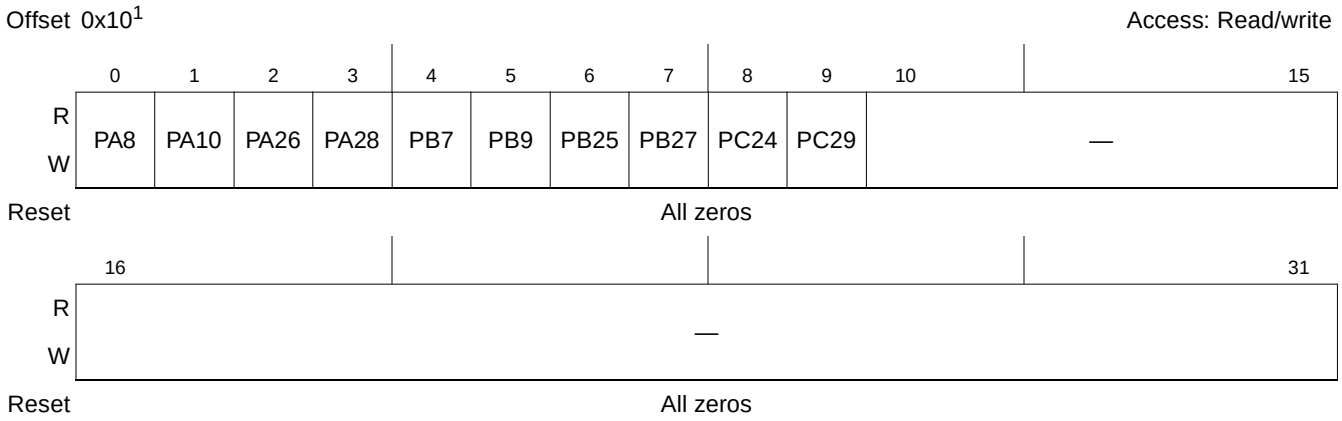


Figure 8-25. QUICC Engine Ports Interrupt Mask Register (CEPIMR)

1 Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See Chapter 2, "Memory Map."

Table 8-33 defines the bit fields of CEPIMR.

Table 8-33. CEPIMR Bit Settings

Bits	Name	Description
0–9	Px[n]	Interrupt mask. Indicates whether an interrupt event is masked or not masked. 0 The input interrupt signal is masked (disabled). 1 The input interrupt signal is not masked (enabled).
10–31	—	Reserved. Should be cleared.

8.5.23 QUICC Engine Ports Interrupt Control Register (CEPICR)

The QUICC Engine ports interrupt control register (CEPICR), shown in Figure 8-26, determines whether the corresponding QUICC Engine port line asserts an interrupt request upon either a high-to-low change or any change on the state of the signal.

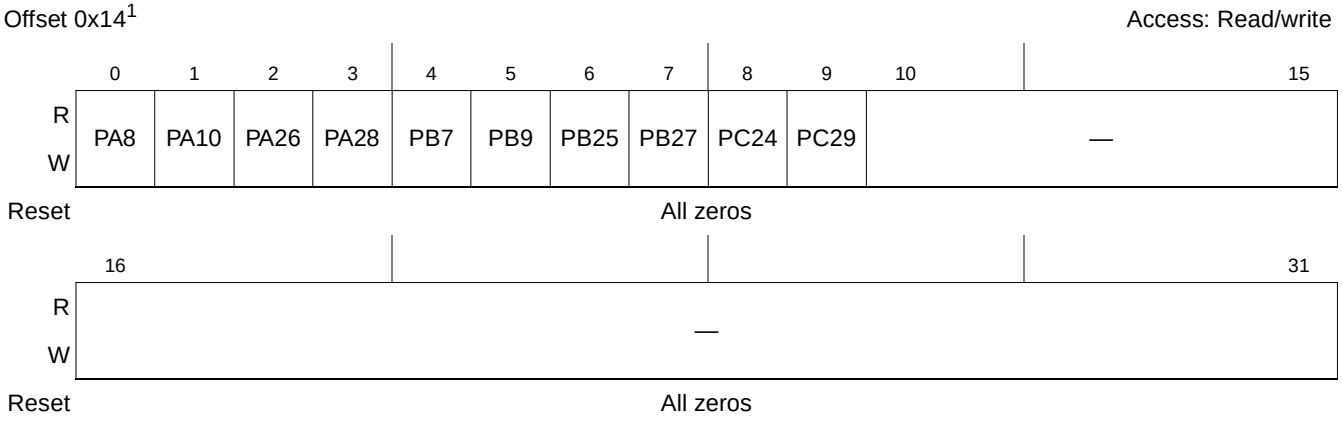


Figure 8-26. QUICC Engine Ports Interrupt Control Register (CEPICR)

1 Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See Chapter 2, "Memory Map."

Table 8-34 defines the bit fields of CEPICR.

Table 8-34. CEPICR Bit Settings

Bits	Name	Description
0–9	Px*n*	Edge detection mode. The corresponding QUICC Engine port line asserts an interrupt request according to the following: 0 Any change on the state of the QUICC Engine port generates an interrupt request. 1 High-to-low change on the QUICC Engine port generates an interrupt request.
10–31	—	Reserved. Should be cleared.

8.6 Functional Description

The following sections describe the types of interrupts, interrupt configurations, and their priorities.

8.6.1 Interrupt Types

The IPIC is responsible for receiving hardware-generated interrupts from different sources (both internal and external) along with prioritizing and delivering them to the CPU for servicing. The interrupt sources are controlled by the IPIC unit and may cause three types of exceptions in the processor core. The $\overline{int}$ signal is the main interrupt output from the IPIC to the processor core and causes the external interrupt exception.The $\overline{cint}$ signal is the critical interrupt output from the IPIC to the processor core and causes the critical external interrupt exception. The $\overline{smi}$ signal is the system management interrupt output from the IPIC to the processor core and causes the system management interrupt exception. The machine check exception is caused by the internal $\overline{mcp}$ signal generated by the IPIC, informing the processor of error conditions, assertion of the external MCP request, and other conditions.

8.6.2 Interrupt Configuration

Figure 8-27 shows the interrupt configuration.

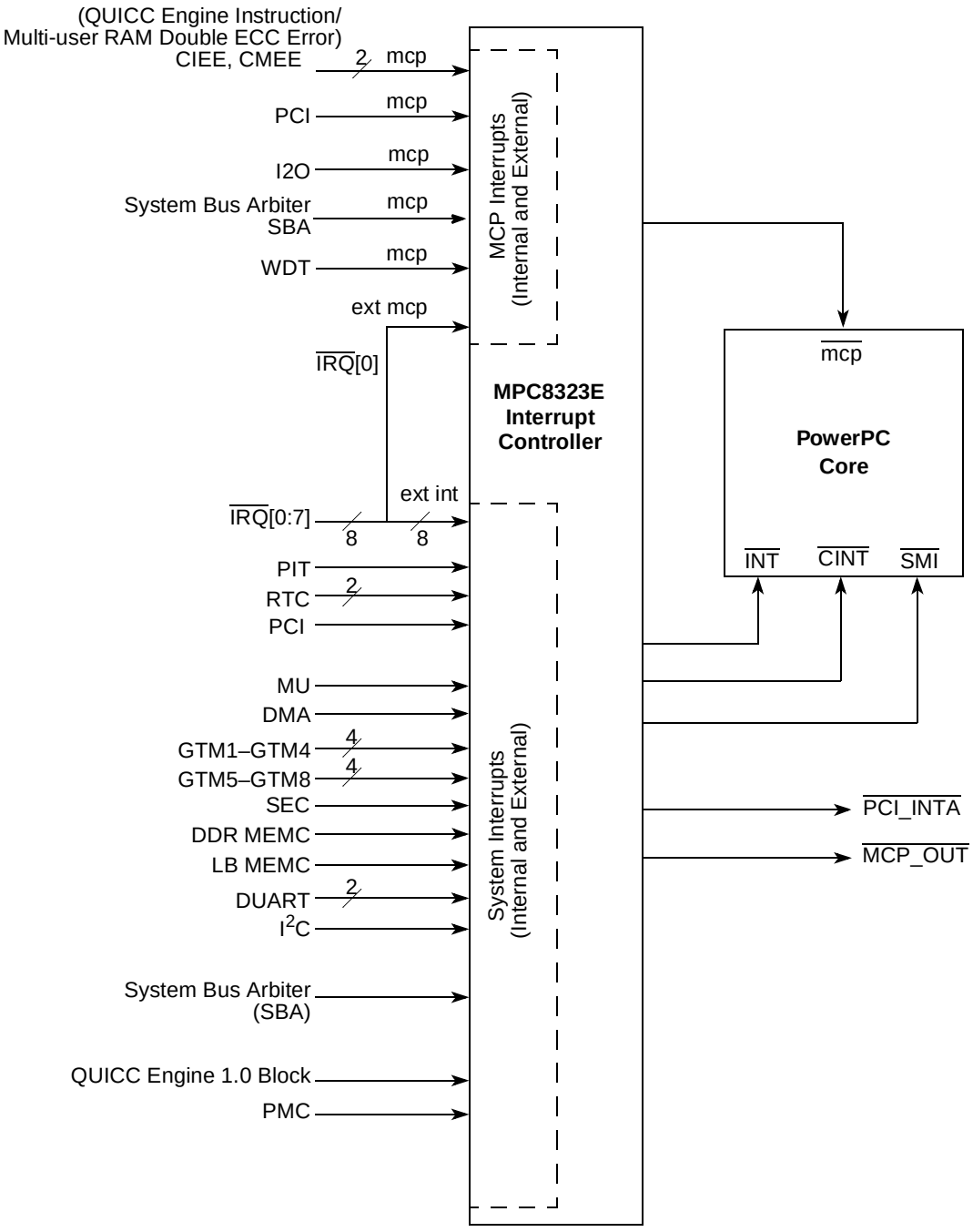


Figure 8-27. Interrupt Structure

The interrupt controller allows masking of each interrupt source. When an unmasked interrupt source is pending in the SIPNR register, the interrupt controller sends an interrupt request to the core. When an interrupt is taken, the interrupt mask bit in the machine state register is cleared to disable further interrupt requests to the PowerPC core until software can handle them.

All interrupt sources are prioritized and bits are set in the system interrupt pending register (SIPNR, SEPNR) as interrupts occur regardless of whether they are masked in the IPIC. The prioritization of the interrupt sources is flexible within the following groups:

- The relative priority of the QE High and QE Low internal interrupt signals can be modified.
- The relative priority of the UART1, UART2, I2C, and SEC internal interrupt signals can be modified.
- The relative priority of the IRQ0, IRQ1, IRQ2, and IRQ3 external interrupts, and RTC SEC internal interrupts can be modified.
- The relative priority of the IRQ4, IRQ5, IRQ6, and IRQ7 external interrupts, and RTC ALR, MU, SBA, and DMA internal interrupts can be modified.
- One interrupt source can be assigned to be the programmable highest priority.

The SIVCR is updated with a 7-bit vector corresponding to the sub-block with the highest current priority.

8.6.3 Internal Interrupts Group Relative Priority

The relative priority in each internal group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC internal interrupt priority registers (SIPRR*x*) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of Table 8-35, ahead of most other interrupt sources. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over Table 8-35 so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

8.6.4 Mixed Interrupts Group Relative Priority

The relative priority between up to four internal and four external interrupts in each group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC mixed interrupt priority registers (SMPRR*x*) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the mixed interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of the priority table, ahead of most other interrupt sources. See Table 8-35 for more information. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over the table so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

8.6.5 Highest Priority Interrupt

In addition to the group relative priority option, SICFR[HPI] can be used to specify one interrupt source as having the highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but is serviced before any other interrupt in Table 8-35.

If the highest priority feature is not used, the IPIC selects the interrupt request in MIXA0 to be the highest priority interrupt and the standard interrupt priority order is used from Table 8-35. SICFR[HPI] can be updated dynamically to allow the user to change a normally low priority source into a high priority-source for a period as needed.

8.6.6 Interrupt Source Priorities

Each of the IPIC's internal and external interrupt sources can independently assert one interrupt request to the core. Table 8-35 shows the prioritization of these interrupt sources. As described in previous sections, flexibility exists in the relative ordering of the interrupts, but, in general, relative priorities are as shown. A single interrupt priority number is associated with each table entry.

Table 8-35. Interrupt Source Priority Levels

Priority	Interrupt Source Description
1	Highest
2	MIXA0 (Grouped/Spread)
3	MIXA1 (Grouped)
4	MIXA2 (Grouped)
5	MIXA3 (Grouped)
6	MIXB0 (Spread)
7–10	Reserved
11	MIXA1 (Spread)
12–15	Reserved
16	MIXB0 (Grouped)
17	MIXB1 (Grouped)
18	MIXB2 (Grouped)
19	MIXB3 (Grouped)
20	MIXB1 (Spread)
21	SYSA0 (Grouped)
22	SYSA1 (Grouped)
23	SYSA2 (Grouped)
24	SYSA3 (Grouped)
25	MIXA2 (Spread)
26	SYSA4 (Grouped)
27	SYSA5 (Grouped)
28	SYSA6 (Grouped)
29	SYSA7 (Grouped)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
30	MIXA4 (Grouped)
31	MIXA5 (Grouped)
32	MIXA6 (Grouped)
33	MIXA7 (Grouped)
34	MIXB2 (Spread)
35–38	Reserved
39	MIXA3 (Spread)
40–43	Reserved
44	MIXB4 (Grouped)
45	MIXB5 (Grouped)
46	MIXB6 (Grouped)
47	MIXB7 (Grouped)
48	MIXB3 (Spread)
49	SYSD0 (Grouped)
50	SYSD1 (Grouped)
51	SYSD2 (Grouped)
52	SYSD3 (Grouped)
53	MIXA4 (Spread)
54	SYSD4 (Grouped)
55	SYSD5 (Grouped)
56	SYSD6 (Grouped)
57	SYSD7 (Grouped)
58	MIXB4 (Spread)
59	GTM4
60	Reserved
61	SYSA0 (Spread)
62	GTM8
63	Reserved
64	SYSD0 (Spread)
65	Reserved
66	QE Ports
67	MIXA5 (Spread)
68	Reserved
69	Reserved
70	SYSA1 (Spread)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
71	DDR
72	Reserved
73	SYSD1 (Spread)
74	Reserved
75	LBC
76	MIXB5 (Spread)
77	GTM2
78	Reserved
79	SYSA2 (Spread)
80	GTM6
81	Reserved
82	SYSD2 (Spread)
83	Reserved
84	PMC
85	MIXA6 (Spread)
86	Reserved
87	Reserved
88	SYSA3 (Spread)
89	Reserved
90	Reserved
91	SYSD3 (Spread)
92	Reserved
93	Reserved
94	MIXB6 (Spread)
95	GTM3
96	Reserved
97	SYSA4 (Spread)
98	GTM7
99	Reserved
100	SYSD4 (Spread)
101	Reserved
102	Reserved
103	MIXA7 (Spread)
104	Reserved
105	Reserved
106	SYSA5 (Spread)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
107	Reserved
108	Reserved
109	SYSD5 (Spread)
110	Reserved
111	Reserved
112	MIXB7 (Spread)
113	GTM1
114	Reserved
115	SYSA6 (Spread)
116	GTM5
117	Reserved
118	SYSD6 (Spread)
119	Reserved
120	Reserved
121	Reserved
122	Reserved
123	SYSA7 (Spread)
124	Reserved
125	Reserved
126	SYSD7 (Spread)
127	Reserved
128	Reserved

8.6.7 Masking Interrupt Sources

By programming the system interrupt mask registers, SIMSR*x* and SEMSR, the user can mask interrupt requests to the core. Each SIMSR*x* and SEMSR bit corresponds to an interrupt source. To enable an interrupt, set the corresponding SIMSR or SEMSR bit. When a masked interrupt source has a pending interrupt request, the corresponding SIPNR*x* or SEMSR bit is set, even though the interrupt is not generated to the core. The user can mask all interrupt sources to implement a polling interrupt servicing scheme.

When an interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that particular block. Table 8-35 shows which interrupt sources have multiple interrupting events.

Figure 8-28 shows an example of how the masking occurs, using a DDR block as an example.

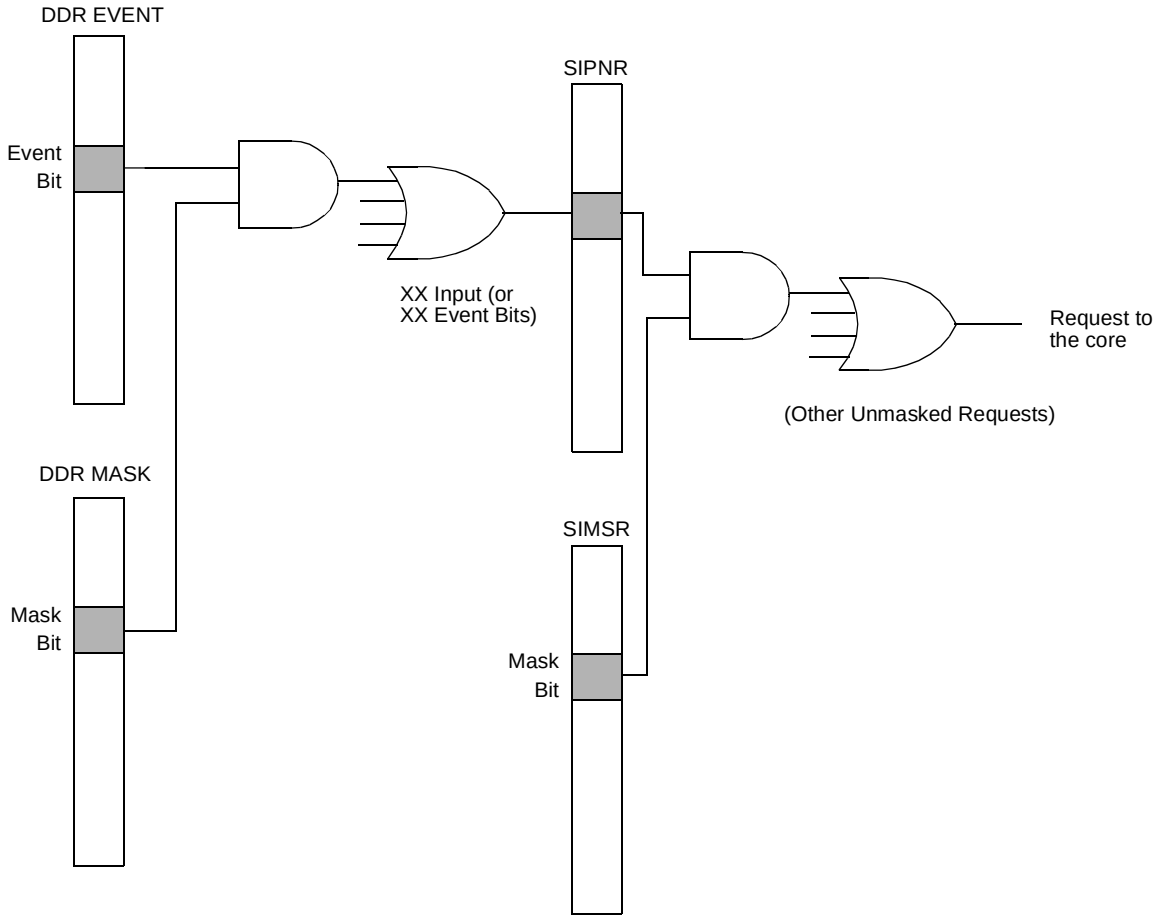


Figure 8-28. DDR Interrupt Request Masking

8.6.8 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority according to Table 8-35. The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by interrupt handler software reading SIVCR. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt in response to a read of SIVCR. Table 8-6 lists the encodings for the seven low-order bits of the interrupt vector.

8.6.9 Machine Check Interrupts

The PIC supports the non-maskable machine check interrupts. When an error interrupt signal is received, the interrupt controller indicates the source by setting the corresponding SERSR bit. These sources are listed in Table 8-22.

8.6.10 QUICC Engine Ports Interrupts

The QUICC Engine ports interrupts are a special case of external interrupt requests, which are specifically related to the QUICC Engine block. Each such QUICC Engine ports external interrupt may be generated upon detection of a high-to-low change on the external signal or upon any change on the external signal (note that this is not the same as for normal $\overline{\text{IRQ}}$ external interrupt signals). All of the QUICC Engine ports interrupts are managed in three registers CEPIER, CEPIMR & CEPICR. If any of the QUICC Engine ports interrupts is pending and not masked, an internal 'QE Ports' event is generated and this is handled through the SIPNR_L register. The specific QUICC Engine ports lines that have this capability are detailed in Section 8.5.21, "QUICC Engine Ports Interrupt Event Register (CEPIER)," and in Table 8-36 below. This feature is specifically useful for pins that can function as modem control signals $\overline{\text{CTS}}$ or $\overline{\text{CD}}$.

Table 8-36. QUICC Engine Ports Interrupt Lines

QUICC Engine Port	$\overline{\text{CTS}}$ / $\overline{\text{CD}}$ Functionality	QUICC Engine Port	$\overline{\text{CTS}}$ / $\overline{\text{CD}}$ Functionality
PA[8]	UCC1:$\overline{\text{CD}}$	PA[28]	UCC2:$\overline{\text{CTS}}$
PA[10]	UCC1:$\overline{\text{CTS}}$	PC[24]	UCC5:$\overline{\text{CD}}$
PA[26]	UCC2:$\overline{\text{CD}}$	PC[29]	UCC5:$\overline{\text{CTS}}$

Chapter 9
DDR Memory Controller

9.1 Introduction

The fully programmable DDR SDRAM controller supports most JEDEC standard x8, x16, or x32 DDR2 and DDR memories available. In addition, unbuffered and registered DRAM modules are supported. However, mixing different memory types or unbuffered and registered DRAM modules in the same system is not supported. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features support rapid system debug.

NOTE

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

Figure 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.

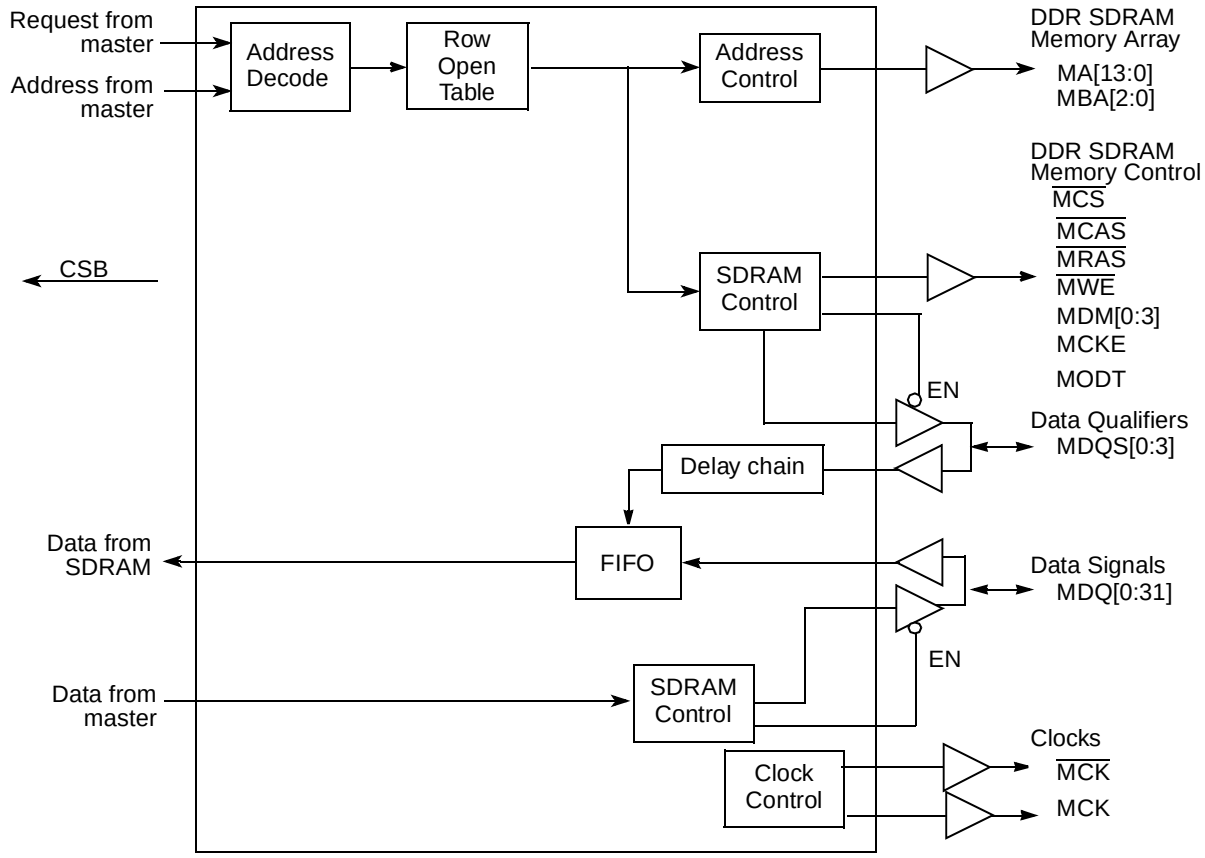


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR2 and DDR SDRAM
- 32--bit SDRAM for DDR and DDR2
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
  - One physical bank (chip select)
  - 64-Mbit to 2-Gbit devices depending on internal device configuration with x8/x16/x32 data ports (no direct x4 support)
  - Unbuffered and registered DRAM modules
- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence
- Automatic DRAM data initialization
- Support for up to eight posted refreshes

- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management

9.2.1 Modes of Operation

The DDR memory controller supports the following modes:

- 32-byte cache line wrap mode
- 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 by setting CS0_CONFIG[AP_0_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.

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 device hardware specification has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

Table 9-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
MDQ[0:31]	Data bus	All zeros	32	I/O
MDQS[0:3]	Data strobes	All zeros	4	I/O
$\overline{\text{MCAS}}$	Column address strobe	One	1	O
MA[13:0]	Address bus	All zeros	14	O
MBA[2:0]	Logical bank address	All zeros	3	O
$\overline{\text{MCS}}$	Chip select	One	1	O
$\overline{\text{MWE}}$	Write enable	One	1	O
$\overline{\text{MRAS}}$	Row address strobe	One	1	O
MDM[0:3]	Data mask	All zeros	4	O
MCK	DRAM clock output	Zero	1	O

Table 9-1. DDR Memory Interface Signal Summary (continued)

Name	Function/Description	Reset	Pins	I/O
$\overline{\text{MCK}}$	DRAM clock output (complement)	One	1	O
MCKE	DRAM clock enable	Zero	1	O
MODT	DRAM on-die termination external control.	Zero	1	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)
msb	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR)[1]
	MA9	A9
	MA8	A8 (alternate AP for DDR)[2]
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

1 Auto-precharge for DDR signaled on A10 when DDR_SDRAM_CFG[PCHB8] = 0

2 Auto-precharge for DDR signaled on A8 when DDR_SDRAM_CFG[PCHB8] = 1

9.3.2 Detailed Signal Descriptions

The following sections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

9.3.2.1 Memory Interface Signals

Table 9-3 describes the DDR controller memory interface signals.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDQ[0:31]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven coincident with corresponding data strobes (MDQS) signal. High impedance—No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represents the state of data being driven by the external DDR SDRAMs.
		Timing	Assertion/Negation—The DDR SDRAM drives data during a READ transaction. High impedance—No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.
MDQS[0:3]	I/O	Data strobes. Inputs with read data, outputs with write data.	
	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is transmitted and driven low when negative capture data is transmitted. 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 9-24 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 received and driven low when negative capture data is received. Centered in the data eye for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 9-24 for byte lane assignments.
		Timing	Assertion/Negation—If a READ command is registered at clock edge *n*, and the latency is programmed in TIMING_CFG_1[CASLAT] to be *m* clocks, data strobes at the DRAM assert coincident with the data on clock edge *n + m*. See the JEDEC DDR SDRAM specification for more information.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MA[13:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[13:0] carry 14 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		State Meaning	Asserted/Negated—Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See Table 9-28 and Table 9-29for a complete description of the mapping of these signals.
		Timing	Assertion/Negation—The address is always driven when the memory controller is enabled. It is valid when a transaction is driven to DRAM (when $\overline{\text{MCS}}$ is active). High impedance—When the memory controller is disabled
MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	
		State Meaning	Asserted/Negated—Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. Table 9-28 and Table 9-29describes 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 9-30 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 9.4.1.4, "DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)," Section 9.4.1.5, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)," Section 9.4.1.6, "DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)," and Section 9.4.1.3, "DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)." High impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is disabled.
$\overline{\text{MRAS}}$	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 9-30 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—Assertion and negation timing is directed by the values described in Section 9.4.1.4, "DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)," Section 9.4.1.5, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)," Section 9.4.1.6, "DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)," and Section 9.4.1.3, "DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)." High impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is disabled.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{MCS}}$	O	Chip select. One chip select 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 (CS0_BNDS)," and Section 9.4.1.2, "Chip Select Configuration (CS0_CONFIG)." The DDR controller asserts $\overline{\text{MCS}}$ 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_0–TIMING_CFG_3. 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 9-30 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 disabled.
MDM[0:3]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. MDM0 corresponds to the most significant byte (MSB). Table 9-24 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.
MODT	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. MODT represents the on-die termination for the associated data, data masks, and data strobes.	
		State Meaning	Asserted/Negated—Represents the ODT driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the CS0_CONFIG[ODT_RD_CFG] and CS0_CONFIG[ODT_WR_CFG] fields. High impedance—Always driven.

9.3.2.2 Clock Interface Signals

Table 9-4 contains the detailed descriptions of the clock signals of the DDR controller.

Table 9-4. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MCK, $\overline{\text{MCK}}$	O	DRAM clock output and its complement. See Section 9.5.4.1, "Clock Distribution." Enabled by the MCKENR*n* register. See Section 4.5.3.1, "MCK Enable Register (MCKENR)," for details.	
		State Meaning	Asserted/Negated—The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		Timing	Assertion/Negation—Timing is controlled by the DDR_CLK_CNTL register at offset 0x130.

Table 9-4. Clock Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MCKE	O	Clock enable. Output signal used as the clock enable to the SDRAM. MCKE can be negated to stop clocking the DDR SDRAM. The MCKE signals should be connected to the same rank of memory as the corresponding $\overline{\text{MCS}}$ and MODT signals. For example, MCKE[0] should be connected to the same rank of memory as $\overline{\text{MCS}}$[0] and MODT[0].	
		State Meaning	Asserted—Clocking to the SDRAM is enabled. Negated—Clocking to the SDRAM is disabled and the SDRAM should ignore signal transitions on MCK or $\overline{\text{MCK}}$. MCK/$\overline{\text{MCK}}$ are don't cares while MCKE is negated.
		Timing	Assertion/Negation—Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Are asserted again when exiting dynamic power management or self refresh. High impedance—Always driven.

9.3.2.3 Debug Signals

The debug signals MSRCID[0:4] and MDVAL have no function in normal DDR controller operation. A detailed description of these signals can be found in Section 5.3.2.6, "Debug Configuration."

9.4 Memory Map/Register Definition

Table 9-5 shows the register memory map for the DDR memory controller.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 9-5. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
DDR Memory Controller—Block Base Address 0x0_2000				
0x000	CS0_BNDS—Chip select memory bounds	R/W	0x0000_0000	9.4.1.1/9-9
0x080	CS0_CONFIG—Chip select configuration	R/W	0x0000_0000	9.4.1.2/9-10
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.3/9-11
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-12
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-14
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-16
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-18
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.8/9-21

Table 9-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.9/9-22
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.10/9-23
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.11/9-24
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.12/9-26
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.13/9-26
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-27
0x140–0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	9.4.1.15/9-27
0x150–0xBF4	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x*nnnn_nnnn*[1]	9.4.1.16/9-28
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00*nn*_00*nn*[1]	9.4.1.17/9-28
0xE00–0xE58	Reserved	—	—	—

1 Implementation-dependent reset values are listed in specified section/page.

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 (CS0_BNDS)

The chip select bounds register (CS0_BNDS) define the starting and ending address of the memory space that corresponds to the chip select. Note that the size specified in CS0_BNDS should equal the size of physical DRAM. Also, note that EA must be greater than or equal to SA.

CS0_BNDS are shown in Figure 9-2.

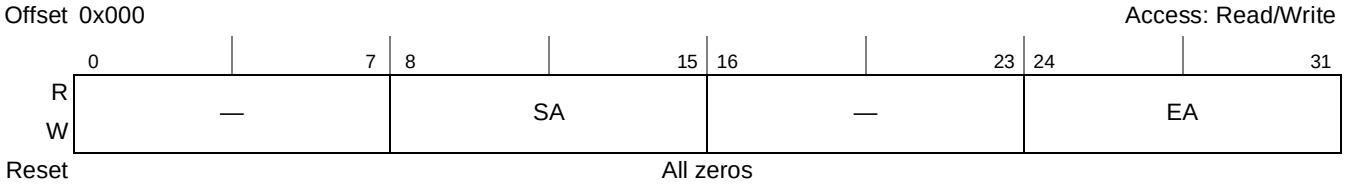


Figure 9-2. Chip Select Bounds Registers (CS0_BNDS)

Table 9-6 describes the CS0_BNDS register fields.

Table 9-6. CS0_BNDS Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SA	Starting address for the chip select (bank). This value is compared against the 8 msbs of the 32-bit address.
16–23	—	Reserved
24–31	EA	Ending address for the chip select (bank). This value is compared against the 8 msbs of the 32-bit address.

9.4.1.2 Chip Select Configuration (CS0_CONFIG)

The chip select configuration (CS0_CONFIG) registers shown in Figure 9-3 enable the DDR chip select and set the number of row and column bits used for the chip select. This register should be loaded with the correct number of row and column bits for each SDRAM. Because CS0_CONFIG[ROW_BITS_CS_0, COL_BITS_CS_0] establish address multiplexing, the user should take great care to set these values correctly.

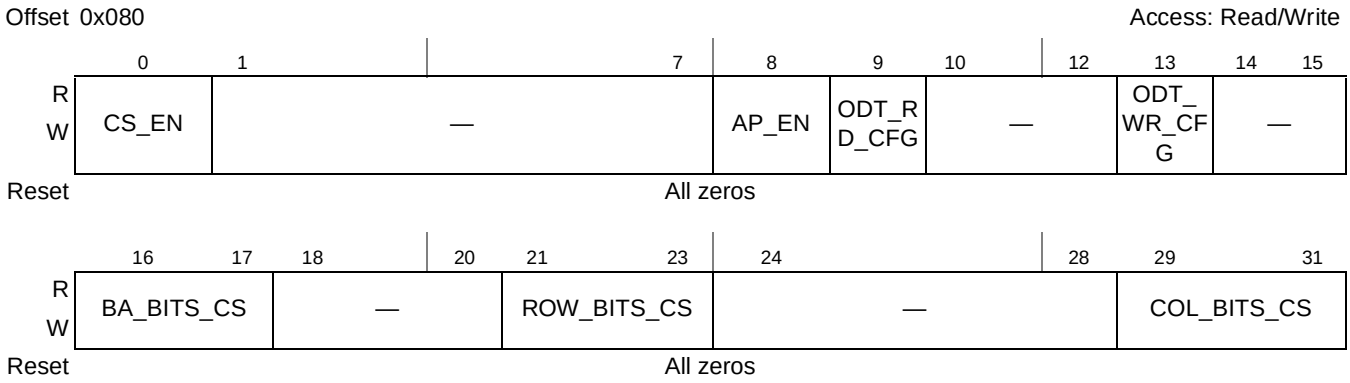


Figure 9-3. Chip Select Configuration Register (CS0_CONFIG)

Table 9-7 describes the CS0_CONFIG register fields.

Table 9-7. CS0_CONFIG Field Descriptions

Bits	Name	Description
0	CS_EN	Chip select enable 0 Chip select is not active 1 Chip select is active and assumes the state set in CS0_BNDS.
8	AP_EN	Chip select auto-precharge enable 0 Chip select is only auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select always issues an auto-precharge for read and write transactions.
9	ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should only be used with DDR2 memories; for such memories, ODT_RD_CFG should be cleared, and ODT_WR_CFG should be set. 0 Never assert ODT for reads 1 Assert ODT only during reads to chip select

Table 9-7. CS0_CONFIG Field Descriptions (continued)

Bits	Name	Description
10–12	—	Reserved
13	ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT _WR_CFG to be enabled. ODT should only be used with DDR2 memories; for such memories, ODT_RD_CFG should be cleared, and ODT_WR_CFG should be set. 0 Never assert ODT for writes 1 Assert ODT only during writes to chip select
14–15	—	Reserved
16–17	BA_BITS_CS	Number of bank bits for SDRAM on chip select. These bits correspond to the sub-bank bits driven on MBA*n* in Table 9-28 and Table 9-29. 00 2 logical bank bits 01 3 logical bank bits 10–11 Reserved
18–20	—	Reserved
21–23	ROW_BITS_CS	Number of row bits for SDRAM on chip select. See Table 9-28 and Table 9-29 for details. 000 12 row bits 001 13 row bits 010 14 row bits 011–111 Reserved
24–28	—	Reserved
29–31	COL_BITS_CS	Number of column bits for SDRAM on chip select. For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100–111 Reserved

9.4.1.3 DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

DDR SDRAM timing configuration register 3, shown in Figure 9-4, sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

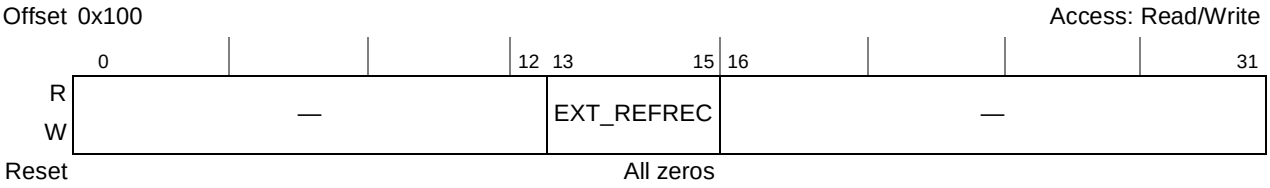


Figure 9-4. DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

Table 9-8 describes TIMING_CFG_3 fields.

Table 9-8. TIMING_CFG_3 Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13–15	EXT_REFREC	Extended refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain a 7bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7bit value of the refresh recovery. t_{RFC} = {EXT_REFREC || REFREC} + 8, such that t_{RFC} is calculated as follows: 000 0 clocks 001 16 clocks 010 32 clocks 011 48 clocks 100 64 clocks 101 80 clocks 110 96 clocks 111 112 clocks
16–31	—	Reserved, should be cleared.

9.4.1.4 DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

DDR SDRAM timing configuration register 0, shown in Figure 9-5, sets the number of clock cycles between various SDRAM control commands.

Offset 0x104 Access: Read/Write

	0 1 2 3	4 5 6 7	8	9 11	12	13 15	16 19	20 23	24 27	28 31
R W	RWT	WRT	RRT	WWT	—	ACT_PD_EXIT	—	PRE_PD_EXIT	—	ODT_PD_EXIT	—	MRS_CYC
Reset	0 0 0 0	0 0 0 0	0	0 0 0 1	0	0 0 0 1	0 0 0 0	0 0 0 1	0 0 0 0	0 1 0 1

Figure 9-5. DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

Table 9-9 describes TIMING_CFG_0 fields.

Table 9-9. TIMING_CFG_0 Field Descriptions

Bits	Name	Description
0–1	RWT	Read-to-write turnaround (t_{RTW}). Specifies how many extra cycles are added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller uses a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller determines the read-to-write turnaround as CL – WL + BL/2 + 2. In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks

Table 9-9. TIMING_CFG_0 Field Descriptions (continued)

Bits	Name	Description
2–3	WRT	Write-to-read turnaround. Specifies how many extra cycles are added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller uses a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller determines the write-to-read turnaround as WL – CL + BL/2 + 1. In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
4–5	RRT	Read-to-read turnaround. Specifies how many extra cycles are added between reads to different chip selects. As a default, 3 cycles are required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
6–7	WWT	Write-to-write turnaround. Specifies how many extra cycles are added between writes to different chip selects. As a default, 2 cycles are required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
8	—	Reserved, should be cleared.
9–11	ACT_PD_EXIT	Active powerdown exit timing (t_{XARD} and t_{XARDS}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 000 Reserved 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.
13–15	PRE_PD_EXIT	Precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
16–19	—	Reserved, should be cleared.

Table 9-9. TIMING_CFG_0 Field Descriptions (continued)

Bits	Name	Description
20–23	ODT_PD_EXIT	ODT powerdown exit timing (t_{AXPD}). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. 0000 0 clock 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
24–27	—	Reserved, should be cleared.
28–31	MRS_CYC	Mode register set cycle time (t_{MRD}). Specifies the number of cycles that must pass after a Mode Register Set command until any other command. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000–1111Reserved

9.4.1.5 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in Figure 9-6, sets the number of clock cycles between various SDRAM control commands.

Offset 0x108 Access: Read/Write

	0	1–3	4–7	8	9–11	12–15	16–19	20	21–23	24	25–27	28	29–31
R W	—	PRETOACT	ACTTOPRE	—	ACTTORW	CASLAT	REFREC	—	WRREC	—	ACTTOACT	—	WRTORD
Reset	All zeros												

Figure 9-6. DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

Table 9-10 describes TIMING_CFG_1 fields.

Table 9-10. TIMING_CFG_1 Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.

Table 9-10. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
1–3	PRETOACT	Precharge-to-activate interval (t_{RP}). Determines the number of clock cycles from a precharge command until an activate or refresh command is allowed. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
4–7	ACTTOPRE	Activate to precharge interval (t_{RAS}). Determines the number of clock cycles from an activate command until a precharge command is allowed. 0000 16 clocks 0001 17 clocks 0010 18 clocks 0011 19 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 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
12–15	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 9.4.1.8, “DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2).”) 0000 Reserved 0001 1 clock 0010 1.5 clocks 0011 2 clocks 0100 2.5 clocks 0101 3 clocks 0110 3.5 clocks 0111 4 clocks 1000 4.5 clocks 1001 5 clocks 1010 5.5 clocks 1011 6 clocks 1100 6.5 clocks 1101 7 clocks 1110 7.5 clocks 1111 8 clocks

Table 9-10. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
16–19	REFREC	Refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7-bit value of the refresh recovery, such that t_{RFC} is calculated as follows: t_{RFC} = {EXT_REFREC || REFREC} + 8. 0000 8 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 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
24	—	Reserved, should be cleared.
25–27	ACTTOACT	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 5 clocks 110 6 clocks 111 7 clocks
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

9.4.1.6 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in Figure 9-7, sets the clock delay to data for writes.

Offset 0x10C — Access: Read/Write

	0	1–3	4–8	9	10–12	13–15	16–18	19–21	22	23–25	26–31
R W	—	ADD_LAT	CPO	—	WR_LAT	—	RD_TO_PRE	WR_DATA_DELAY	—	CKE_PLS	FOUR_ACT
Reset	All zeros										

Figure 9-7. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)

Table 9-11 describes the TIMING_CFG_2 fields.

Table 9-11. TIMING_CFG_2 Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. (DDR2-specific) 000 0 clocks 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 Reserved 111 Reserved
4–8	CPO[1]	$\overline{\text{MCAS}}$-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. For these decodings, “READ_LAT” is equal to the CAS latency plus the additive latency. 00000 READ_LAT + 1 00001 Reserved 00010 READ_LAT 00011 READ_LAT + 1/4 00100 READ_LAT + 1/2 00101 READ_LAT + 3/4 00110 READ_LAT + 1 00111 READ_LAT + 5/4 01000 READ_LAT + 3/2 01001 READ_LAT + 7/4 01010 READ_LAT + 2 01011 READ_LAT + 9/4 01100 READ_LAT + 5/2 01101 READ_LAT + 11/4 01110 READ_LAT + 3 01111 READ_LAT + 13/4 10000 READ_LAT + 7/2 10001 READ_LAT + 15/4 10010 READ_LAT + 4 10011 READ_LAT + 17/4 10100 READ_LAT + 9/2 10101 READ_LAT + 19/4 10110–11111 Reserved
9	—	Reserved
10–12	WR_LAT	Write latency. Note that the total write latency for DDR2 is equal to WR_LAT + ADD_LAT; the write latency for DDR1 is 1. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
13–15	—	Reserved
16–18	RD_TO_PRE	Read to precharge (t_{RTP}). For DDR2, with a non-zero ADD_LAT value, takes a minimum of ADD_LAT + t_{RTP} cycles between read and precharge. For DDR1 with burst length of 4, must be set to 010; for DDR1 with burst length of 8, must be set to 100. 000 Reserved 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101–111 Reserved

Table 9-11. TIMING_CFG_2 Field Descriptions (continued)

Bits	Name	Description
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. See Section 9.5.7, "DDR SDRAM Write Timing Adjustments," for details. 000 0 clock delay 001 1/4 clock delay 010 1/2 clock delay 011 3/4 clock delay 100 1 clock delay 101 5/4 clock delay 110 3/2 clock delay 111 Reserved
22	—	Reserved
23–25	CKE_PLS	Minimum CKE pulse width (t_{CKE})Can be set to 001 for DDR1. 000 Reserved 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101–111 Reserved
26–31	FOUR_ACT	Window for four activates (t_{FAW}). This is applied to DDR2 with eight logical banks only. Must be set to 000001 for DDR1. 000000 Reserved 000001 1 cycle 000010 2 cycles 000011 3 cycles 000100 4 cycles ... 010011 19 cycles 010100 20 cycles 010101–111111 Reserved

1 For CPO decodings other than 00000 and 11111, 'READ_LAT' is rounded up to the next integer value.

9.4.1.7 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in Figure 9-8, enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

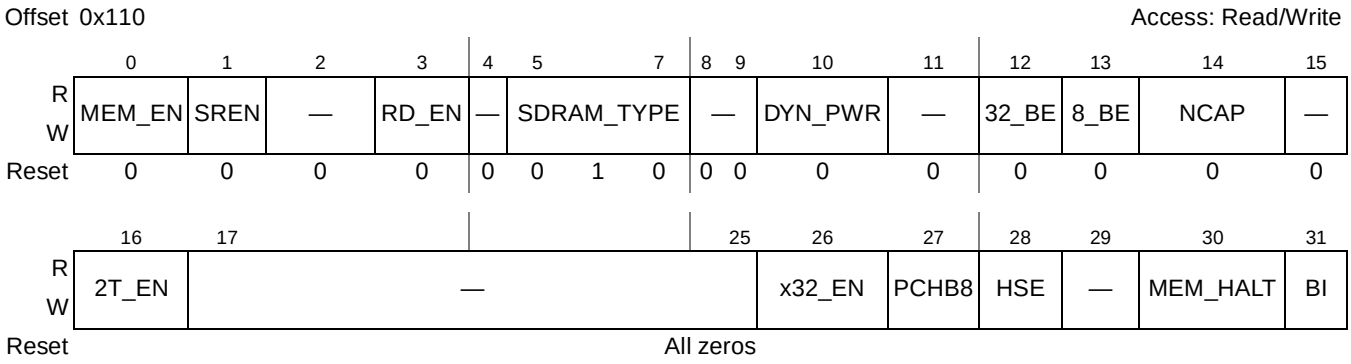


Figure 9-8. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

Table 9-12 describes the DDR_SDRAM_CFG fields.

Table 9-12. DDR_SDRAM_CFG Field Descriptions

Bits	Name	Description
0	MEM_EN	DDR SDRAM interface logic enable. 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1	SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep. 1 SDRAM self refresh is enabled during sleep.
2	—	Reserved. Must be cleared.
3	RD_EN	Registered DRAM module enable. Specifies the type of DRAM module used in the system. 0 Indicates unbuffered DRAM modules. 1 Indicates registered DRAM modules. **Note:** RD_EN and 2T_EN must not both be set at the same time.
4	—	Reserved
5–7	SDRAM_TYPE	Type of SDRAM device to be used. This field is used when issuing the automatic hardware initialization sequence to DRAM through Mode Register Set and Extended Mode Register Set commands. Default value is 010 designating DDR1 SDRAM. 000–001Reserved 010 DDR1 SDRAM 011 DDR2 SDRAM 100 Reserved 101 Reserved 110 Reserved 111 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	—	Reserved
12	32_BE	32-bit bus enable. 0 Reserved 1 32-bit bus is used. Software MUST set this bit.
13	8_BE	8-beat burst enable. 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface. **Note:** DDR1 (SDRAM_TYPE = 010) must use 8-beat bursts when using 32-bit bus mode (32_BE = 1) and 4-beat bursts when using 64-bit bus mode; DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode
14	NCAP	Non-concurrent auto-precharge. Some older DDR DRAMs do not support concurrent auto precharge. If one of these devices is used, then this bit needs to be set if auto precharge is used. 0 DRAMs in system support concurrent auto-precharge. 1 DRAMs in system do not support concurrent auto-precharge.
15	—	Reserved

Table 9-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
16	2T_EN	Enable 2T timing. 0 1T timing is enabled. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the second cycle. **Note:** RD_EN and 2T_EN must not both be set at the same time.
17–25	—	Reserved
26	x32_EN	x32 enable. 0 Either x8 or x16 discrete DRAM chips are used. In this mode, each data byte has a dedicated corresponding data strobe. 1 x32 discrete DRAM chips are used. In this mode, DQS0 is used to capture DQ[0:31].
27	PCHB8	Precharge bit 8 enable. 0 MA[10] is used to indicate the auto-precharge and precharge all commands. 1 MA[8] is used to indicate the auto-precharge and precharge all commands. If x32_EN is cleared, then PCHB8 should be cleared as well.
28	HSE	Global half-strength override Sets I/O driver impedance to half strength. This impedance is used by the address/command, data, and clock impedance values. This bit should be cleared if using automatic hardware calibration. 0 I/O driver impedance is configured to full strength. 1 I/O driver impedance is configured to half strength.
29	—	Reserved
30	MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller does not accept any new data read/write transactions to DDR SDRAM until the bit is cleared again. This can be used when bypassing initialization and forcing MODE REGISTER SET commands through software. 0 DDR controller accepts new transactions. 1 DDR controller finishes any remaining transactions, and then it remains halted until this bit is cleared by software.
31	BI	Bypass initialization 0 DDR controller cycles through initialization routine based on SDRAM_TYPE 1 Initialization routine is bypassed. Software is responsible for initializing memory through DDR_SDRAM_MODE2 register. If software is initializing memory, then the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. Note that the DDR controller does not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self refresh after the controller is enabled.

9.4.1.8 DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2, shown in Figure 9-9, provides more control configuration for the DDR controller.

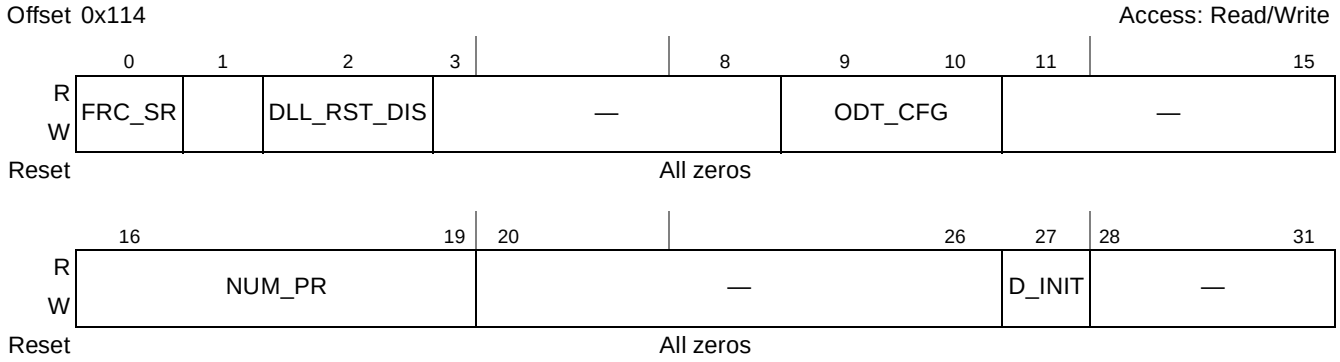


Figure 9-9. DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)

Table 9-13 describes the DDR_SDRAM_CFG_2 fields.

Table 9-13. DDR_SDRAM_CFG_2 Field Descriptions

Bits	Name	Description
0	FRC_SR	Force self refresh 0 DDR controller operates in normal mode. 1 DDR controller enters self-refresh mode.
1	—	Reserved. Should be cleared.
2	DLL_RST_DIS	DLL reset disable. The DDR controller typically issues a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization. 0 DDR controller issues a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller does not issue a DLL reset to the DRAMs when exiting self refresh.
3–8	—	Reserved
9–10	ODT_CFG	ODT configuration. This field defines how ODT is driven to the on-chip IOs. See," which defines the termination value that is used. (DDR2-specific, must be cleared for DDR1) 00 Never assert ODT to internal IOs 01 Assert ODT to internal IOs only during writes to DRAM 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
11–15	—	Reserved.

Table 9-13. DDR_SDRAM_CFG_2 Field Descriptions (continued)

Bits	Name	Description
16–19	NUM_PR	Number of posted refreshes. This determines how many posted refreshes, if any, can be issued at one time. Note that if posted refreshes are used, then this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed such that the maximum t_{ras} specification cannot be violated. For example, some DDR1 SDRAMs are not able to use more than 3 posted refreshes because the required refresh interval could then exceed the maximum constraint for t_{ras}. 0000 Reserved 0001 1 refresh is issued at a time 0010 2 refreshes is issued at a time 0011 3 refreshes is issued at a time ... 1000 8 refreshes is issued at a time 1001–1111 Reserved
20–26	—	Reserved, should be cleared.
27	D_INIT	DRAM data initialization This bit is set by software, and it is cleared by hardware. If software sets this bit before the memory controller is enabled, the controller automatically initializes DRAM after it is enabled. This bit is automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle. 0 There is not data initialization in progress, and no data initialization is scheduled 1 The memory controller initializes memory once it is enabled. This bit remains asserted until the initialization is complete. The value in DDR_DATA_INIT register is used to initialize memory.
28–31	—	Reserved

9.4.1.9 DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register, shown in Figure 9-10, sets the values loaded into the DDR's mode registers.

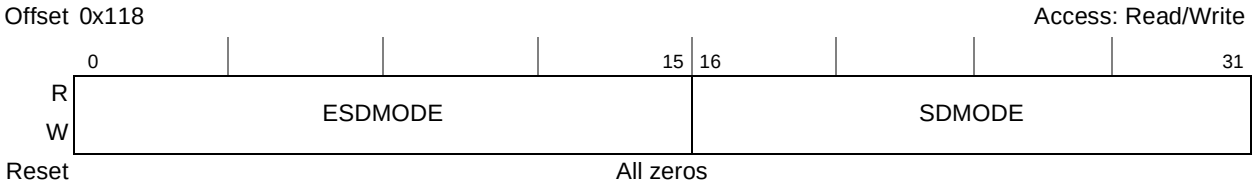


Figure 9-10. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

Table 9-14 describes the DDR_SDRAM_MODE fields.

Table 9-14. DDR_SDRAM_MODE Field Descriptions

Bits	Name	Description
0–15	ESDMODE	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. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in Figure 9-10, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0].
16–31	SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in Figure 9-10, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

9.4.1.10 DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register, shown in Figure 9-11, sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2).

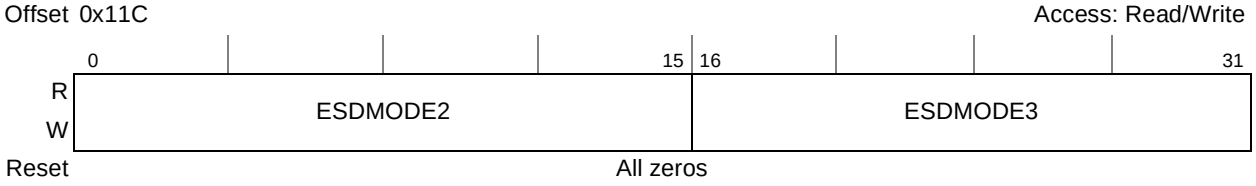


Figure 9-11. DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2)

Table 9-15 describes the DDR_SDRAM_MODE_2 fields.

Table 9-15. DDR_SDRAM_MODE_2 Field Descriptions

Bits	Name	Description
0–15	ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in Figure 9-11, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31	ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in Figure 9-11, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

9.4.1.11 DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register, shown in Figure 9-12, allows the user to carry out the following tasks:

- Issue a mode register set command
- Issue an immediate refresh
- Issue an immediate precharge or precharge all command
- Force the CKE signal to a specific value

Table 9-16 describes the fields of this register. Table 9-17 shows the user how to set the fields of this register to accomplish the above tasks.

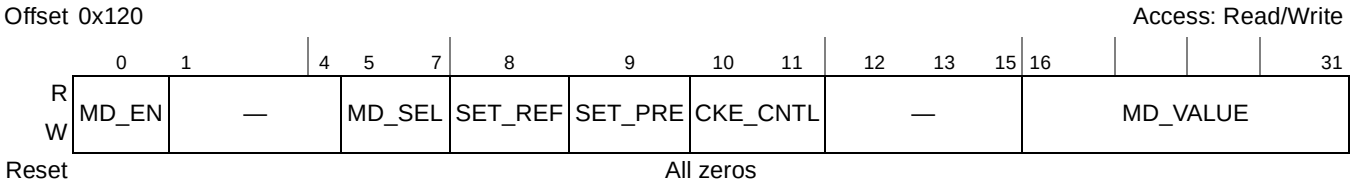


Figure 9-12. DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

Table 9-16 describes the DDR_SDRAM_MD_CNTL fields.

NOTE

Note that MD_EN, SET_REF, and SET_PRE are mutually exclusive; only one of these fields can be set at a time.

Table 9-16. DDR_SDRAM_MD_CNTL Field Descriptions

Bits	Name	Description
0	MD_EN	Mode enable. Setting this bit specifies that valid data in MD_VALUE is ready to be written to DRAM as one of the following commands: • MODE REGISTER SET • EXTENDED MODE REGISTER SET • EXTENDED MODE REGISTER SET 2 • EXTENDED MODE REGISTER SET 3 The specific command to be executed is selected by setting MD_SEL. MD_EN is set by software and cleared by hardware once the command has been issued. 0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.
1–4	—	Reserved
5–7	MD_SEL	Mode register select. MD_SEL specifies one of the following: • During a mode select command, selects the SDRAM mode register to be changed • During a precharge command, selects the SDRAM logical bank to be precharged. A precharge all command ignores this field. • During a refresh command, this field is ignored. Note that MD_SEL contains the value that is presented onto the memory bank address pins (MBA*n*) of the DDR controller. 000 MR 001 EMR 010 EMR2 011 EMR3

Table 9-16. DDR_SDRAM_MD_CNTL Field Descriptions (continued)

Bits	Name	Description
8	SET_REF	Set refresh. Forces an immediate refresh to be issued. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no refresh command needs to be issued. 1 Indicates that a refresh command is ready to be issued.
9	SET_PRE	Set precharge. Forces a precharge or precharge all to be issued. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no precharge all command needs to be issued. 1 Indicates that a precharge all command is ready to be issued.
10–11	CKE_CNTL	Clock enable control. Allows software to clear or set the CKE signal issued to DRAM. Once software has forced the value driven on CKE, that value continues to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller continues to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (such as, self refresh entry/exit, power down entry/exit). 00 CKE signals are not forced by software. 01 CKE signals are forced to a low value by software. 10 CKE signals are forced to a high value by software. 11 Reserved
12–15	—	Reserved
16–31	MD_VALUE	Mode register value. This field, which specifies the value that is presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command. For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant: 0 Issue a precharge command; MD_SEL selects the logical bank to be precharged 1 Issue a precharge all command; all logical banks are precharged

Table 9-17 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 9-17. Settings of DDR_SDRAM_MD_CNTL Fields

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	—
SET_REF	0	1	0	—
SET_PRE	0	0	1	—
MD_SEL	Select mode register. See Table 9-16.	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only bit five is significant. See Table 9-16.	—
CKE_CNTL	0	0	0	See Table 9-16.

9.4.1.12 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in Figure 9-13, sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, the number of DRAM cycles that a page is maintained after it is accessed is provided here.

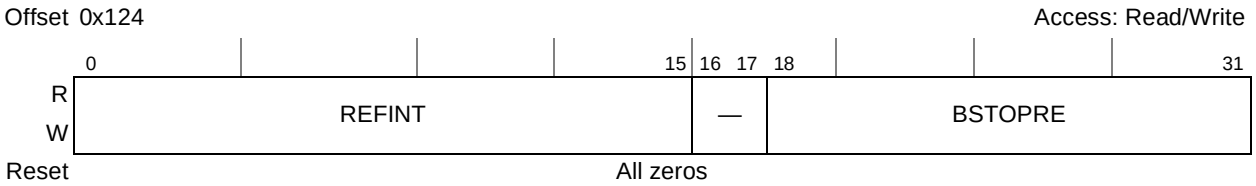


Figure 9-13. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

Table 9-18 describes the DDR_SDRAM_INTERVAL fields.

Table 9-18. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	Description
0–15	REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes are not issued when the REFINT is set to all 0s.
16–17	—	Reserved
18–31	BSTOPRE	Precharge interval. Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

9.4.1.13 DDR SDRAM Data Initialization (DDR_DATA_INIT)

The DDR SDRAM data initialization register, shown in Figure 9-14, provides the value that is used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

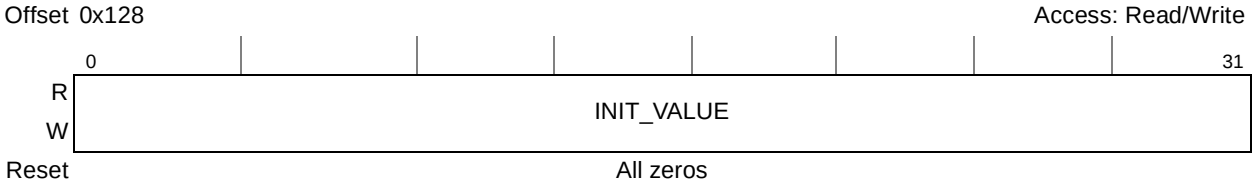


Figure 9-14. DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT)

Table 9-19 describes the DDR_DATA_INIT fields.

Table 9-19. DDR_DATA_INIT Field Descriptions

Bits	Name	Description
0–31	INIT_VALUE	Initialization value. Represents the value that DRAM is initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

9.4.1.14 DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register, shown in Figure 9-15, provides a 1/4-cycle clock adjustment.

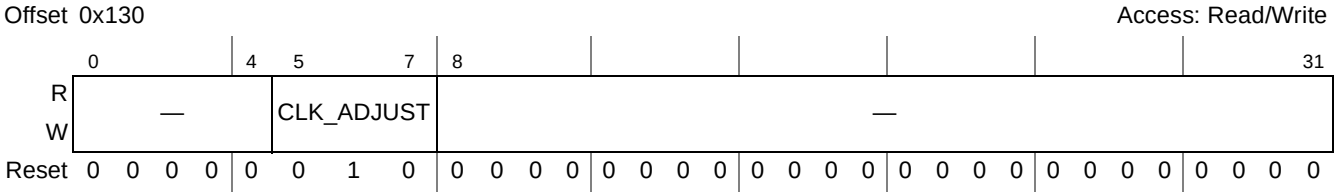


Figure 9-15. DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)

Table 9-20 describes the DDR_SDRAM_CLK_CNTL fields.

Table 9-20. DDR_SDRAM_CLK_CNTL Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	CLK_ADJUST	Clock adjust 000 Clock is launched aligned with address/command 001 Clock is launched 1/4 applied cycle after address/command 010 Clock is launched 1/2 applied cycle after address/command 011 Clock is launched 3/4 applied cycle after address/command 100 Clock is launched 1 applied cycle after address/command 101–111 Reserved
8	—	Reserved, should be set to 0
9–31	—	Reserved

9.4.1.15 DDR Initialization Address (DDR_INIT_ADDR)

The DDR SDRAM initialization address register, shown in Figure 9-16, provides the address that is used for the automatic $\overline{\text{CAS}}$ to preamble calibration after POR.

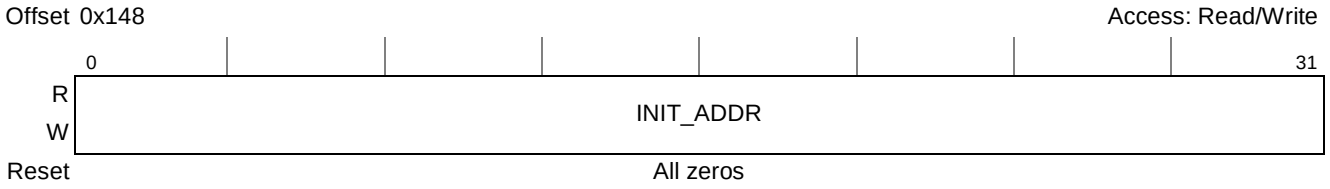


Figure 9-16. DDR Initialization Address Configuration Register (DDR_INIT_ADDR)

Table 9-21 describes the DDR_INIT_ADDR fields.

Table 9-21. DDR_INIT_ADDR Field Descriptions

Bits	Name	Description
0–31	INIT_ADDR	Initialization address. Represents the address that is used for the automatic CAS to preamble calibration at POR.

9.4.1.16 DDR IP Block Revision 1 (DDR_IP_REV1)

The DDR IP block revision 1 register, shown in Figure 9-17, provides read-only fields with the IP block ID, along with major and minor revision information.

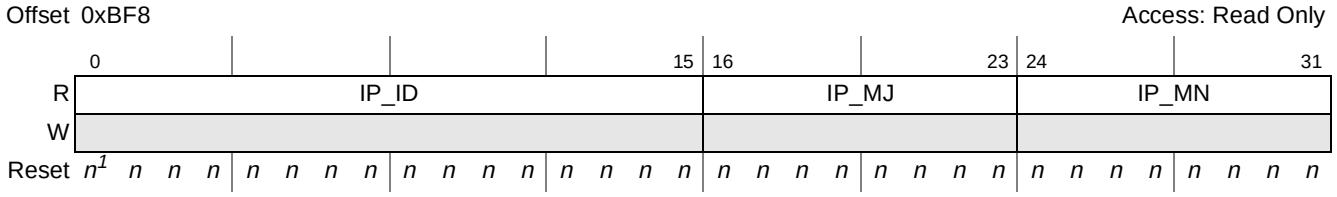


Figure 9-17. DDR IP Block Revision 1 (DDR_IP_REV1)

1 For reset values, see Table 9-22.

Table 9-22 describes the DDR_IP_REV1 fields.

Table 9-22. DDR_IP_REV1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID. For the DDR controller, this value is 0x0002.
16–23	IP_MJ	Major revision. This is currently set to 0x02.
24–31	IP_MN	Minor revision. This is currently set to 0x00.

9.4.1.17 DDR IP Block Revision 2 (DDR_IP_REV2)

The DDR IP block revision 2 register, shown in Figure 9-18, provides read-only fields with the IP block integration and configuration options.

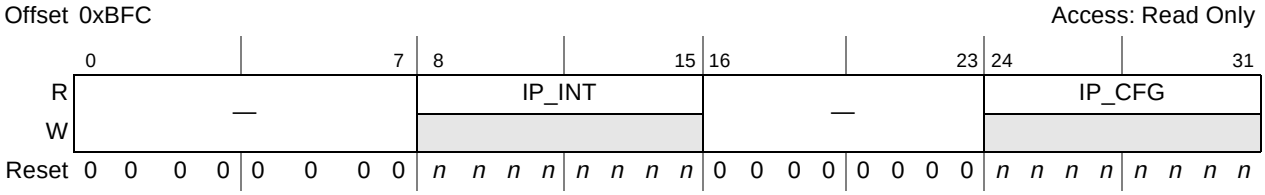


Figure 9-18. DDR IP Block Revision 2 (DDR_IP_REV2)

Table 9-23 describes the DDR_IP_REV2 fields.

Table 9-23. DDR_IP_REV2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

9.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR2 and DDR SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DRAM modules and unbuffered DRAM modules. However, registered DRAM modules cannot be mixed with unbuffered DRAM modules.

Figure 9-19 is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports one physical bank of 32-bit wide memory. Bank sizes up to 512 Mbytes are supported, providing up to a maximum of 512 Mbytes of DDR main memory.

Programmable parameters allow for a variety of memory organizations and timings. The controller allows as many as 8 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

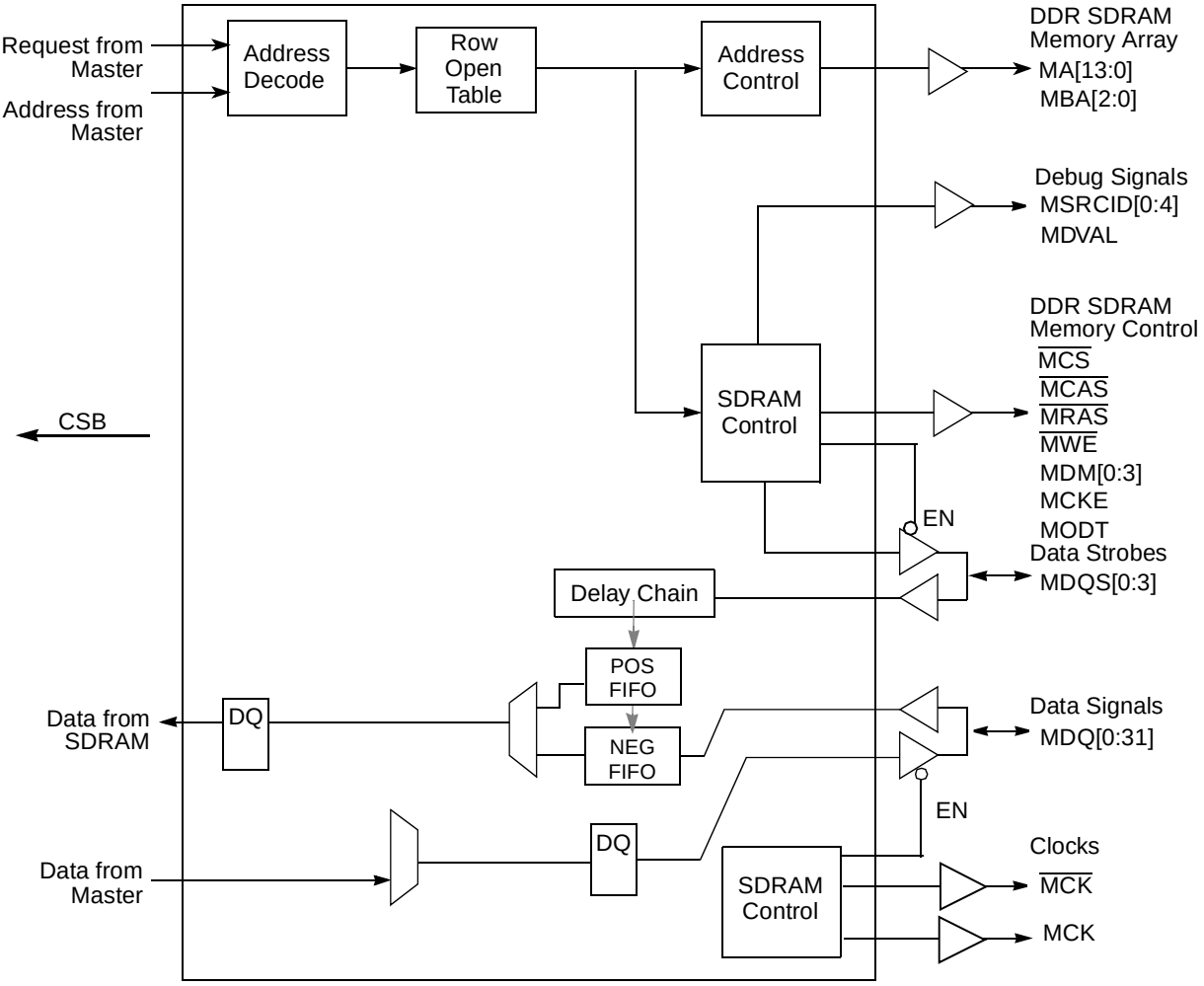


Figure 9-19. DDR Memory Controller Block Diagram

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

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:3]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

The address and command interface is also source synchronous, although 1/4 cycle adjustments are provided for adjusting the clock alignment.

Figure 9-20 shows an example DDR SDRAM configuration with four logical banks.

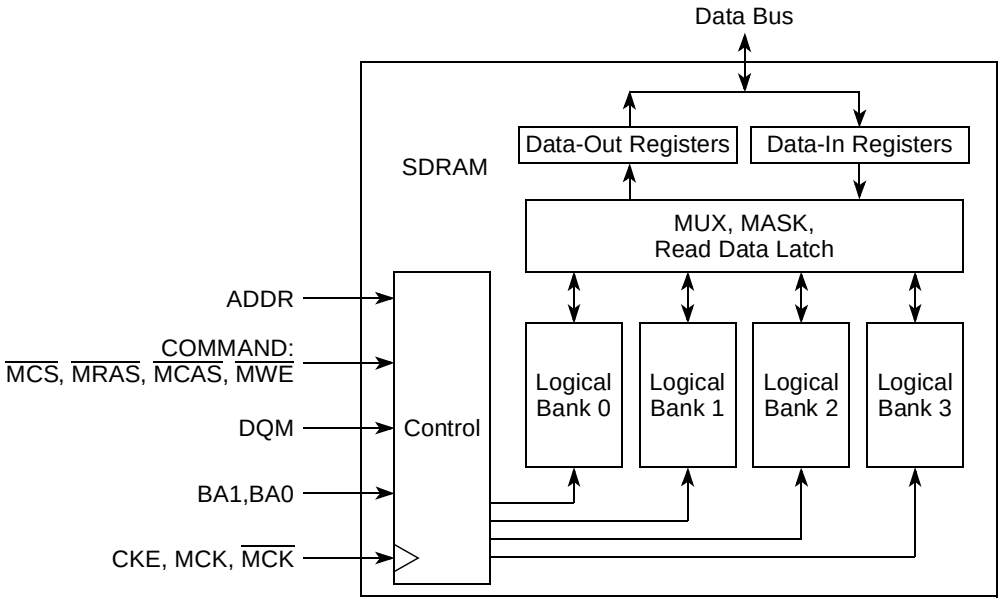


Figure 9-20. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-21 shows some typical signal connections.

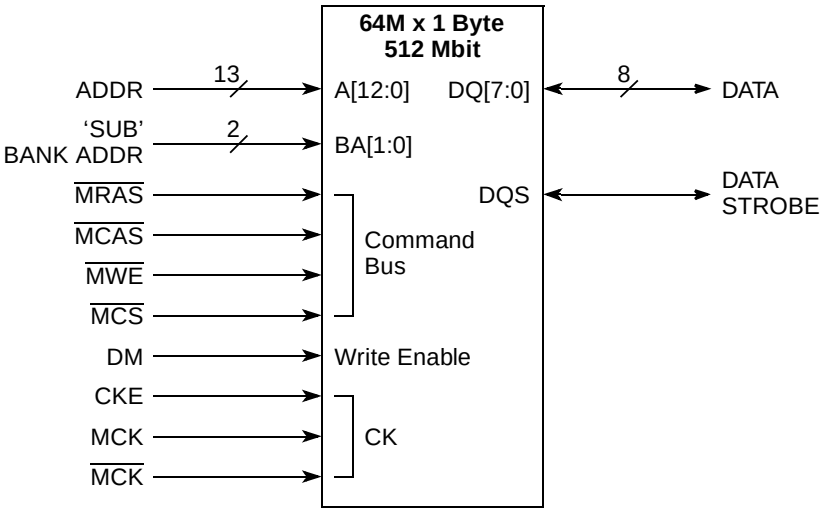
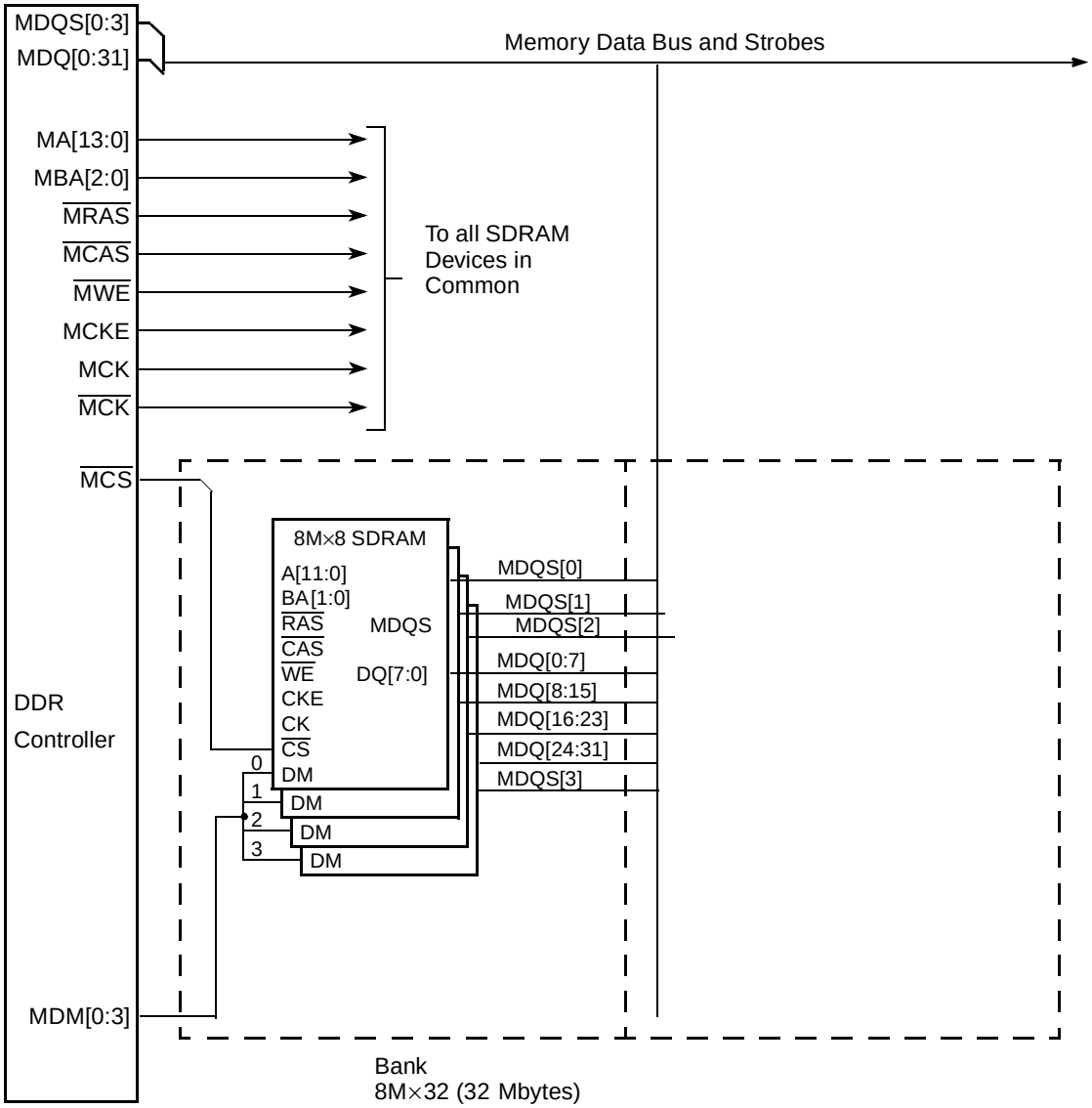


Figure 9-21. Typical DDR SDRAM Interface Signals

Figure 9-22 shows an example DDR SDRAM configuration with one physical bank comprised of four 8M × 8 DDR modules for a total of 32 Mbytes of system memory. Certain address and control lines may require buffering. Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controller drives 14 address pins, but in this example the DDR SDRAM devices use only 12 bits.



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

Figure 9-22. Example 32-Mbyte DDR SDRAM Configuration

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 three logical bank select signals support device densities from 64 Mbits to 2 Gbits. The chip select ($\overline{\text{CS}}$) signal supports one bank 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 bits wide. The DDR memory controller supports physical bank sizes from 16 Mbytes to 512 Mbytes. The physical banks can

be constructed using x8, x16, or x32 memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, and 1 Gbit. Some 2-Gbit devices are supported depending on the internal device configuration. Four data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

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

Table 9-24 shows the DDR memory controller's relationships between data byte lane0–3, MDM[0:3], MDQS[0:3], and MDQ[0:31] when DDR SDRAM memories are used with x8 or x16 devices.

Table 9-24. Byte Lane to Data Relationship

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus
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]

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 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 30 address bits. The physical bank may be configured to provide from 12 to 14 row address bits, plus 2 or 3 logical bank-select bits and from 8–11 column address bits.

Table 9-25 and Table 9-26 describe DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 29 total address bits.

Table 9-25. Supported DDR1 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	32-Bit Bank Size
64 Mbits	8 Mbits x 8	12 x 9 x 2	32 Mbytes
64 Mbits[1]	4 Mbits x 16	12 x 8 x 2	16 Mbytes
128 Mbits	16 Mbits x 8	12 x 10 x 2	64 Mbytes
128 Mbits	8 Mbits x 16	12 x 9 x 2	32 Mbytes
256 Mbits	32 Mbits x 8	13 x 10 x 2	128 Mbytes
256 Mbits	16 Mbits x 16	13 x 9 x 2	64 Mbytes
512 Mbits	64 Mbits x 8	13 x 11 x 2	256 Mbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	128 Mbytes

Table 9-25. Supported DDR1 SDRAM Device Configurations (continued)

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	32-Bit Bank Size
1 Gbits	128 Mbits x 8	14 x 11 x 2	512 Mbytes
1 Gbits	64 Mbits x 16	14 x 10 x 2	256 Mbytes

[1] This configuration is not supported in 16-bit bus mode.

Table 9-26. Supported DDR2 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	32-Bit Bank Size
256 Mbits	32 Mbits x 8	13 x 10 x 2	128 Mbytes
256 Mbits	16 Mbits x 16	13 x 9 x 2	64 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	256 Mbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	128 Mbytes
1 Gbits	128 Mbits x 8	14 x 10 x 3	512 Mbytes
1 Gbits	64 Mbits x 16	13 x 10 x 3	256 Mbytes
2 Gbits	128Mbits x 16	14 x 10 x 3	512 Mbytes

Table 9-27. Supported DDR2 SDRAM Device Configurations—One Physical Bank

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	16-Bit Bank Size
256 Mbits	32 Mbits x 8	13 x 10 x 2	64 Mbytes
256 Mbits	16 Mbits x 16	13 x 9 x 2	32 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	128 Mbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	64 Mbytes
1 Gbits	128 Mbits x 8	14 x 10 x 3	256 Mbytes
1 Gbits	64 Mbits x 16	13 x 10 x 3	128 Mbytes
2 Gbits	256Mbits x 8	14 x 11 x 3	512 Mbytes
2 Gbits	128Mbits x 16	14 x 10 x 3	256 Mbytes

If a transaction request is issued to the DDR memory controller and the address does not lie within the programmed address range, a memory select error is flagged.

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 $\overline{\text{MCS}}$ 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 and Table 9-29 show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[13:0] use MA[13] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR1/DDR2 modes for reads and writes, so the column address can never use MA[10].

Table 9-28. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																												lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30–31
14 x 11 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				11	9	8	7	6	5	4	3	2	1	0
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
13 x 11 x 2	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				11	9	8	7	6	5	4	3	2	1	0
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
13 x 9 x 2	$\overline{\text{MRAS}}$							12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA																				1	0									
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0
12 x 10 x 2	$\overline{\text{MRAS}}$							11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
12 x 9 x 2	$\overline{\text{MRAS}}$								11	10	9	8	7	6	5	4	3	2	1	0											
	MBA																				1	0									
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0
12 x 8 x 2	$\overline{\text{MRAS}}$									11	10	9	8	7	6	5	4	3	2	1	0										
	MBA																					1	0								
	$\overline{\text{MCAS}}$																							7	6	5	4	3	2	1	0

Table 9-29. DDR2 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	30–31
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$							12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0	

9.5.3 JEDEC Standard DDR SDRAM Interface Commands

The following section describes the commands and timings the controller uses when operating in DDR2 or DDR modes.

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

The following DDR SDRAM interface commands (summarized in Table 9-30) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate—Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read—Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.

- Write—Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.
- Refresh (similar to $\overline{\text{MCAS}}$ before $\overline{\text{MRAS}}$)—Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- Mode register set (for configuration)—Allows setting of DDR SDRAM options. These options are: $\overline{\text{MCAS}}$ latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length. $\overline{\text{MCAS}}$ latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide $\overline{\text{MCAS}}$ latency {1,2,3}, some provide $\overline{\text{MCAS}}$ latency {1,2,3,4,5}, and so on). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. A burst length of 8 is supported for DDR1memory only. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4-beat) accesses to the SDRAMs in the memory controller. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory controller after DDR_SDRAM_CFG[MEM_EN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, then the MODE registers can be configured through software through use of the DDR_SDRAM_MD_CNTL register.
- Self refresh (for long periods of standby)—Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller places all logical banks in a precharged state.

Table 9-30. DDR SDRAM Command Table

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

Table 9-30. DDR SDRAM Command Table (continued)

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
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 DDR SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:3].

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 9-31 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-31. DDR SDRAM Interface Timing Intervals

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

Table 9-31. DDR SDRAM Interface Timing Intervals (continued)

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

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_0, TIMING_CFG_1, TIMING_CFG_2, and TIMING_CFG_3 registers as described in Section 9.4.1.4, “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” Section 9.4.1.5, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” Section 9.4.1.6, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and Section 9.4.1.3, “DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)”) and be kept in the DDR memory controller configuration register space.

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

Figure 9-23 through Figure 9-25 show DDR SDRAM timing for various types of accesses; see Figure 9-23 for a single-beat read operation, Figure 9-24 for a single-beat write operation, and Figure 9-25 for a burst-write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads. These figures assume the CLK_ADJUST is set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle (for DDR1).

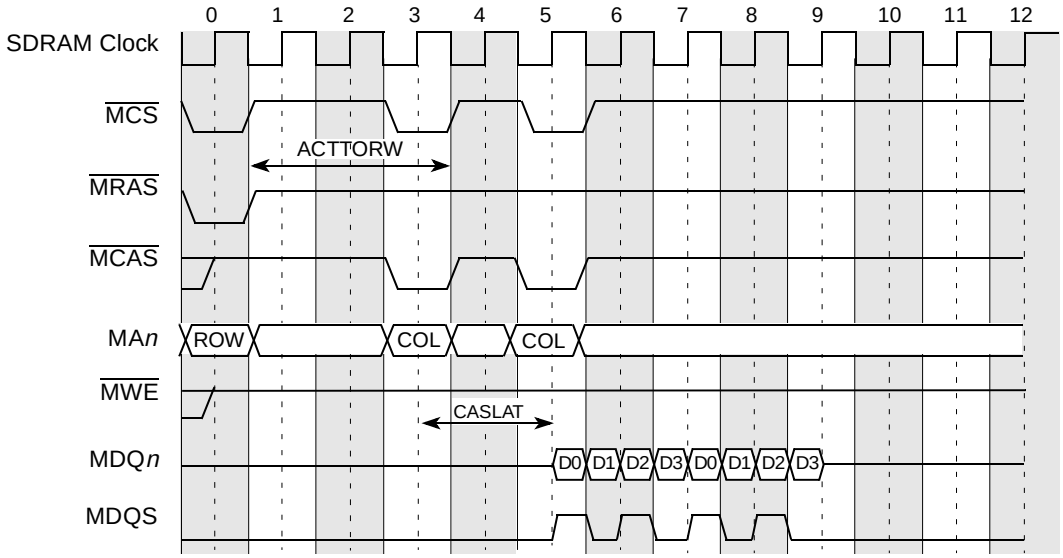


Figure 9-23. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

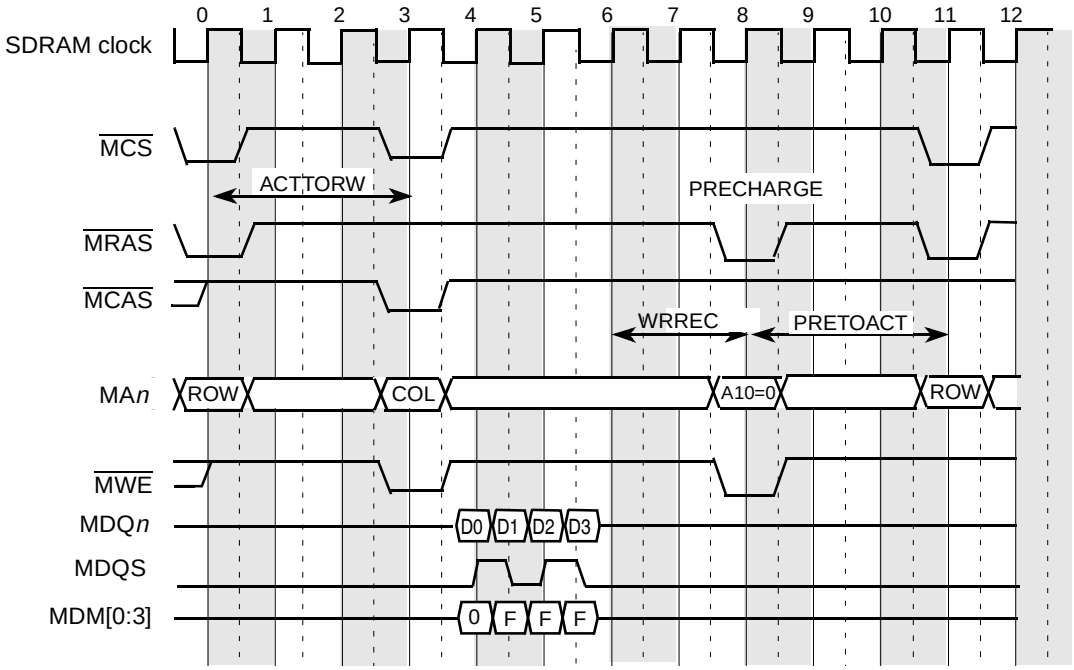


Figure 9-24. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTOR

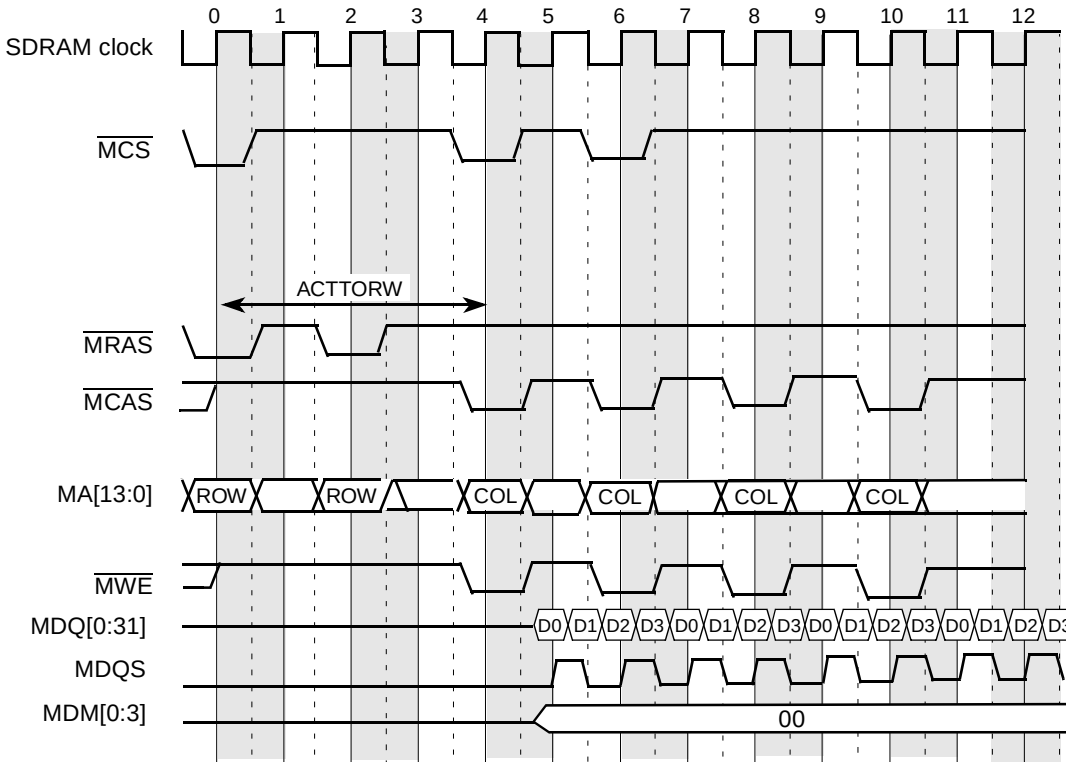


Figure 9-25. DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4

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.
- 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 9-26. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

9.5.5 DDR SDRAM Mode-Set Command Timing

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

Figure 9-27 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

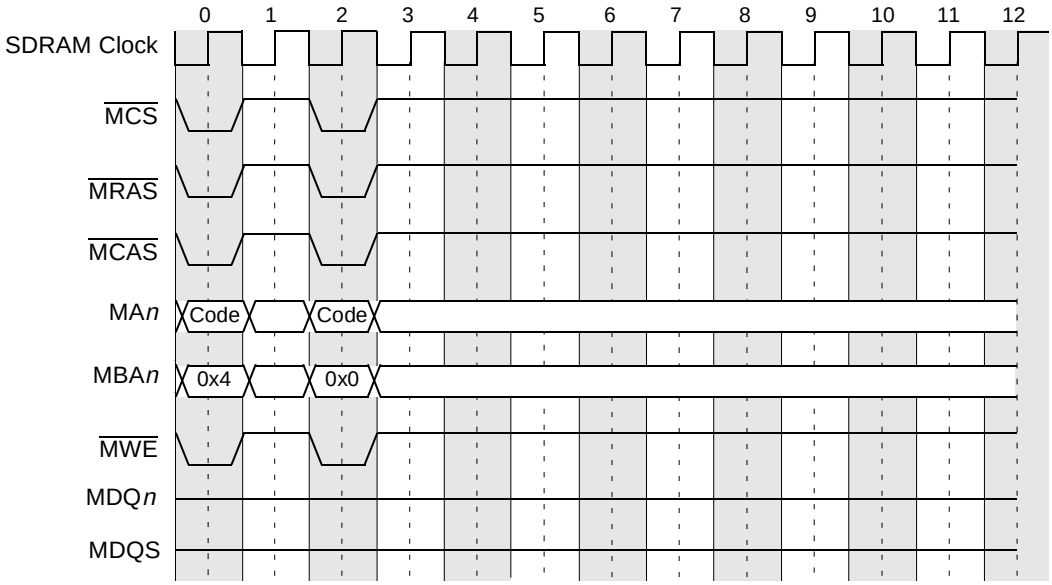


Figure 9-27. DDR SDRAM Mode-Set Command Timing

9.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DRAM modules 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 DRAM modules' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

NOTE

Application system board must assert the reset signal on DDR memory devices until software is able to program the DDR memory controller configuration registers, and must deassert the reset signal on DDR memory devices before DDR_SDRAM_CFG[MEM_EN] is set. This ensures that the DDR memory devices are held in reset until a stable clock is provided and, further, that a stable clock is provided before memory devices are released from reset.

Figure 9-28 shows the registered DDR SDRAM DIMM single-beat write timing.

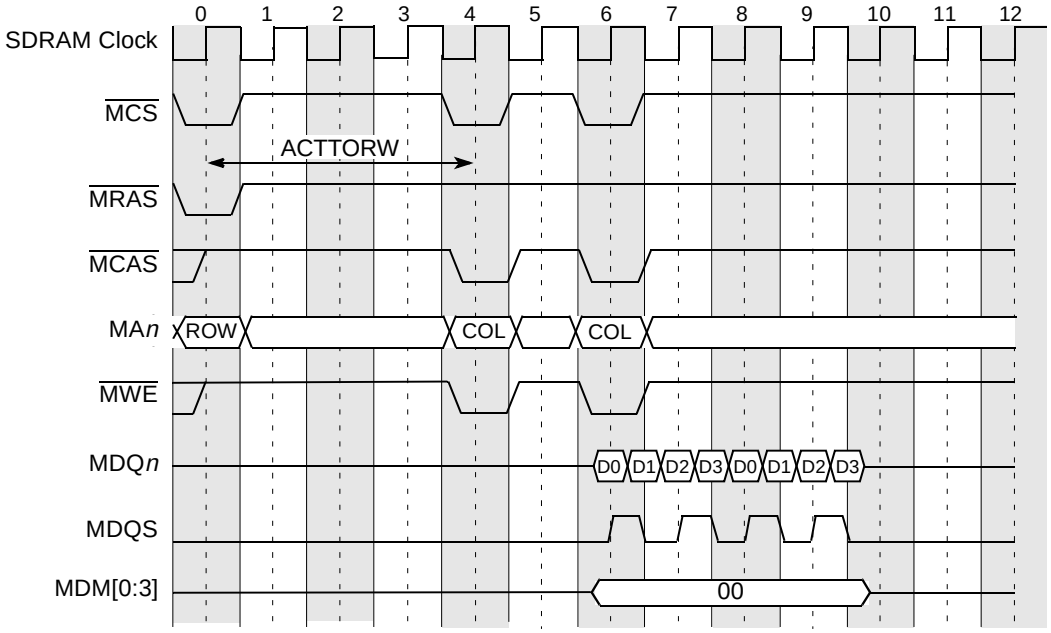


Figure 9-28. Registered DDR SDRAM DIMM Burst Write Timing

9.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0.

Figure 9-29 shows the use of the WR_DATA_DELAY parameter.

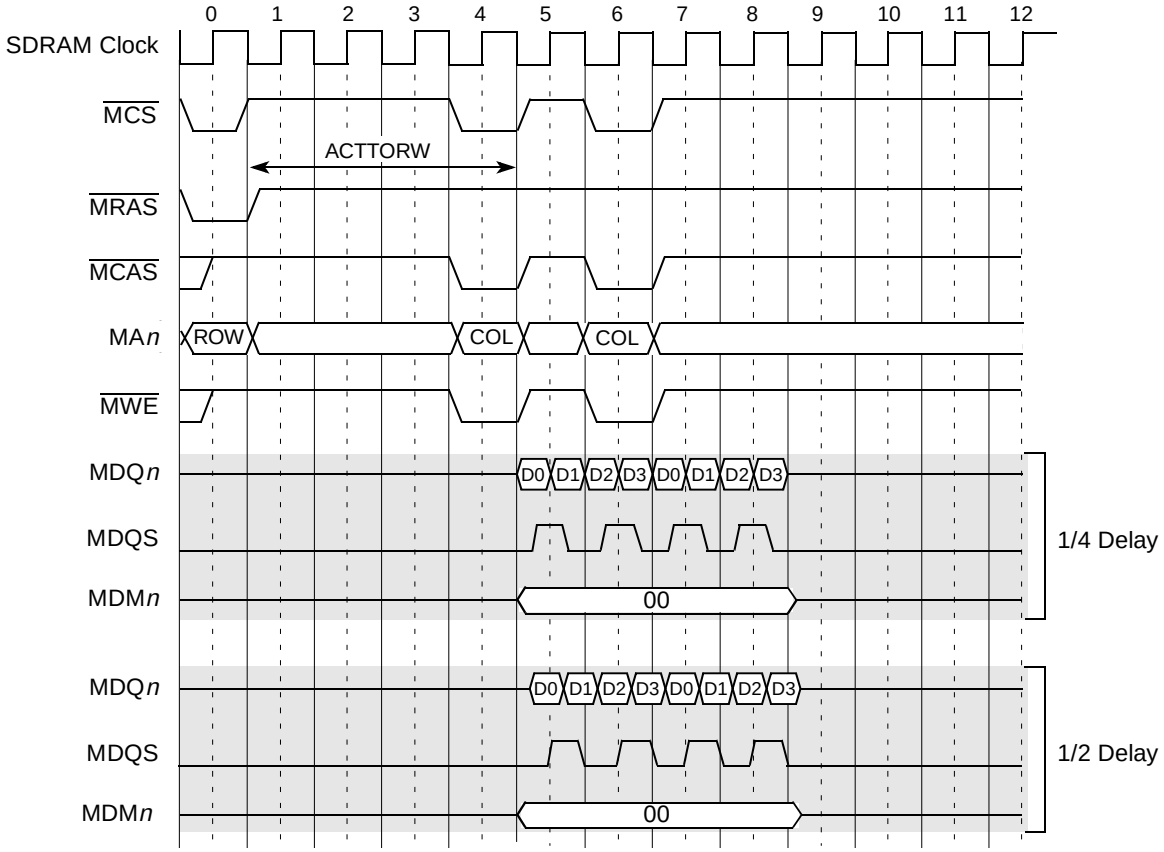


Figure 9-29. Write Timing Adjustments Example for Write Latency = 1

9.5.8 DDR SDRAM Refresh

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

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

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

After the refresh completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC]. In addition, posted refreshes are supported to allow the refresh interval to be set to a larger value.

9.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in Figure 9-30 (TIMING_CFG_1 [REFREC] = 10 in this example).

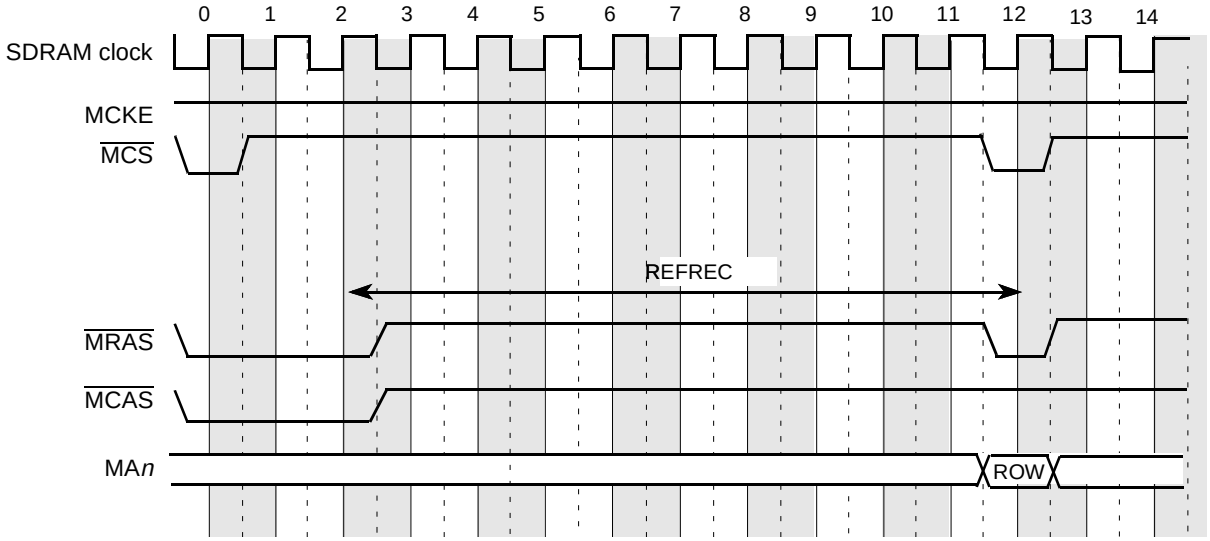


Figure 9-30. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

9.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

Table 9-32 summarizes the refresh types available in each power-saving mode.

Table 9-32. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	—

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in Figure 9-31.

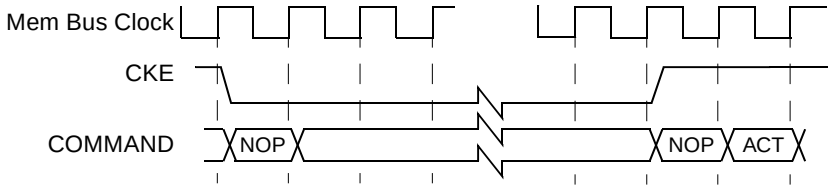


Figure 9-31. DDR SDRAM Power-Down Mode

9.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-32 and Figure 9-33.

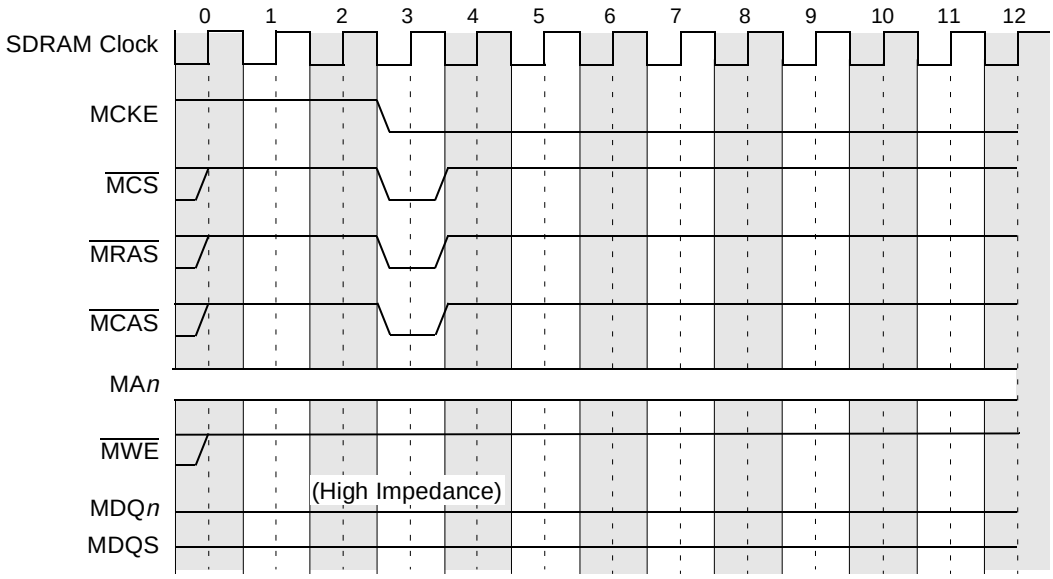


Figure 9-32. DDR SDRAM Self-Refresh Entry Timing

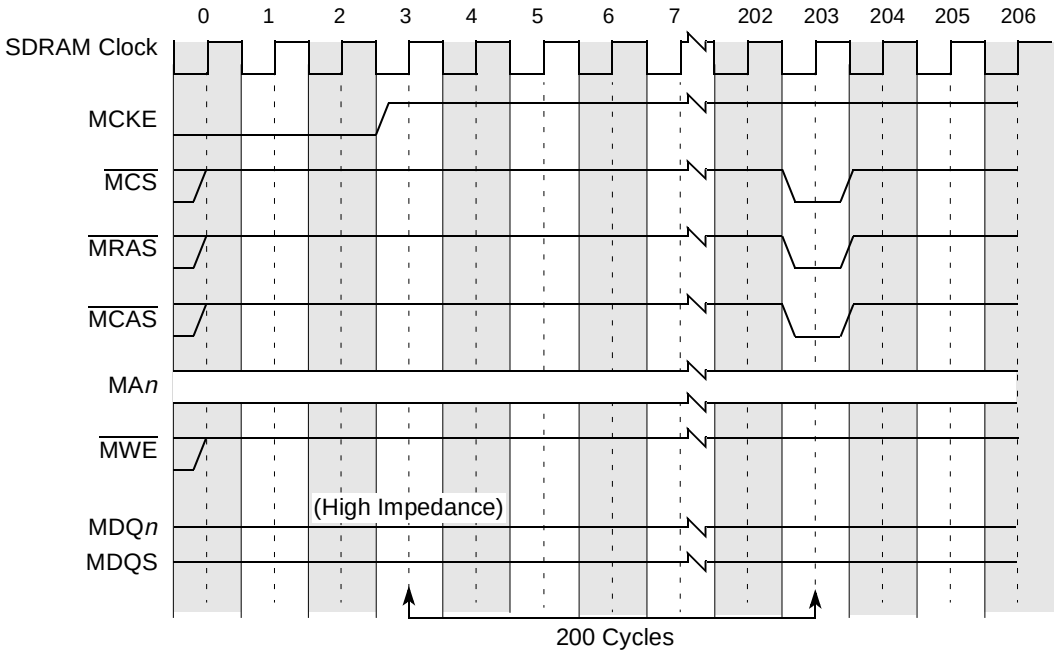


Figure 9-33. DDR SDRAM Self-Refresh Exit Timing

9.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts. For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. The DDR memory controller 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 9-33 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 9-33. 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

1 All underlined **Double**-word offsets are valid for the transaction.

9.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains 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. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CS0_CONFIG[AP_0EN].

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 memory. Then, the DDR memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}$ signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for the bank. Refer to row-address configuration in Section 9.4.1.2, "Chip Select Configuration (CS0_CONFIG)." Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See Section 9.4.1, "Register Descriptions," for more detailed descriptions of the configuration registers. These parameters are shown in Table 9-34.

Table 9-34. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter		Section/page
CS0_BNDS	Chip select memory bounds	SA0 EA0		9.4.1.1/9-9
CS0_CONFIG	Chip select configuration	CS_0_EN AP_0_EN ODT_RD_CFG ODT_WR_CFG	BA_BITS_CS_0 ROW_BITS_CS_0 COL_BITS_CS_0	9.4.1.2/9-10
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC		9.4.1.3/9-11
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT	ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	9.4.1.4/9-12
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT	REFREC WRREC ACTTOACT WRTORD	9.4.1.5/9-14
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE	WR_DATA_DELAY CKE_PLS FOUR_ACT	9.4.1.6/9-16
DDR_SDRAM_CFG	Control configuration	SREN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW	NCAP 2T_EN x32_EN HSE BI	9.4.1.7/9-18
DDR_SDRAM_CFG_2	Control configuration	ODT_CFG	NUM_PR D_INIT	9.4.1.8/9-21
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE		9.4.1.9/9-22

Table 9-34. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/page
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3	9.4.1.10/9-23
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE	9.4.1.12/9-26
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE	9.4.1.13/9-26
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST	9.4.1.14/9-27
DDR_INIT_ADDR	Initialization address	INIT_ADDR	9.4.1.15/9-27

9.6.1 Programming Differences between Memory Types

Depending on the memory type used, certain fields must be programmed differently. Table 9-35 illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 9-35. Programming Differences between Memory Types

Parameter	Description	Differences		Section/page
AP0_EN	Chip Select Auto Precharge Enable	DDR1	Can be used to place chip select in auto precharge mode	9.4.1.2/9-10
		DDR2	Can be used to place chip select in auto precharge mode	
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR1	Should always be set to 000	9.4.1.2/9-10
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select typically not uses ODT when issuing reads to the memory.	
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR1	Should always be set to 000	9.4.1.2/9-10
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT typically is set to assert for the chip select (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	DDR1	Should be set to 0001	9.4.1.4/9-12
		DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{AXPD}.	
PRETOACT	Precharge to Activate Timing	DDR1	Should be set according to the specifications for the memory used (t_{RP})	9.4.1.5/9-14
		DDR2	Should be set according to the specifications for the memory used (t_{RP})	

Table 9-35. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/page
ACTTOPRE	Activate to Precharge Timing	DDR1	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	9.4.1.5/9-14
		DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	
ACTTORW	Activate to Read/Write Timing	DDR1	Should be set according to the specifications for the memory used (t_{RCD})	9.4.1.5/9-14
		DDR2	Should be set according to the specifications for the memory used (t_{RCD})	
CASLAT	CAS Latency	DDR1	Should be set, along with the Extended CAS Latency, to the desired CAS latency	9.4.1.5/9-14
		DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	DDR1	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (t_{RFC})	9.4.1.5/9-14
		DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	
WRREC	Write Recovery	DDR1	Should be set according to the specifications for the memory used (t_{WR})	9.4.1.5/9-14
		DDR2	Should be set according to the specifications for the memory used (t_{WR})	
ACTTOACT	Activate *A* to Activate *B*	DDR1	Should be set according to the specifications for the memory used (t_{RRD})	9.4.1.5/9-14
		DDR2	Should be set according to the specifications for the memory used (t_{RRD})	
WRTORD	Write to Read Timing	DDR1	Should be set according to the specifications for the memory used (t_{WTR})	9.4.1.5/9-14
		DDR2	Should be set according to the specifications for the memory used (t_{WTR})	
ADD_LAT	Additive Latency	DDR1	Should be set to 000	9.4.1.6/9-16
		DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR1	Should be set to 001	9.4.1.6/9-16
		DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency if 5 cycles, then this field should be set to 100 (4 cycles).	

Table 9-35. Programming Differences between Memory Types (continued)

Parameter	Description	Differences		Section/page
RD_TO_PRE	Read to Precharge Timing	DDR1	Should be set to 010 if burst length is 4 and 100 if burst length is 8	9.4.1.6/9-16
		DDR2	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR1	Can be set to 001	9.4.1.6/9-16
		DDR2	Should be set according to the specifications for the memory used (t_{CKE})	
FOUR_ACT	Four Activate Window	DDR1	Should be set to 00001	9.4.1.6/9-16
		DDR2	Should be set according to the specifications for the memory used (t_{FAW}). Only applies to eight logical banks.	
RD_EN	Registered DIMM Enable	DDR1	If registered DRAM modules are used, then this field should be set to 1	9.4.1.7/9-18
		DDR2	If registered DRAM modules are used, then this field should be set to 1	
8_BE	8-beat burst enable	DDR1	If 8-beat bursts are desired, then this field should be set to 1	9.4.1.7/9-18
		DDR2	Should be set to 0	
2T_EN	2T Timing Enable	DDR1	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	9.4.1.7/9-18
		DDR2	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	
ODT_CFG	ODT Configuration	DDR1	Should be set to 00	9.4.1.8/9-21
		DDR2	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
BSTOPR	Burst To Precharge Interval	DDR1	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	9.4.1.12/9-26
		DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

9.6.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs must elapse after

DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and the chip select is enabled) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] is not set, the DDR memory controller conducts an automatic initialization sequence to the memory, which follows the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

Chapter 10
Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers and contains a functional description of the general-purpose chip-select machine (GPCM) and user-programmable machines (UPMs) of the LBC. Finally, it includes initialization and applications information sections with many specific examples of its use.

10.1 Introduction

Figure 10-1 is a functional block diagram of the LBC, which supports two interfaces: GPCM and UPM controller.

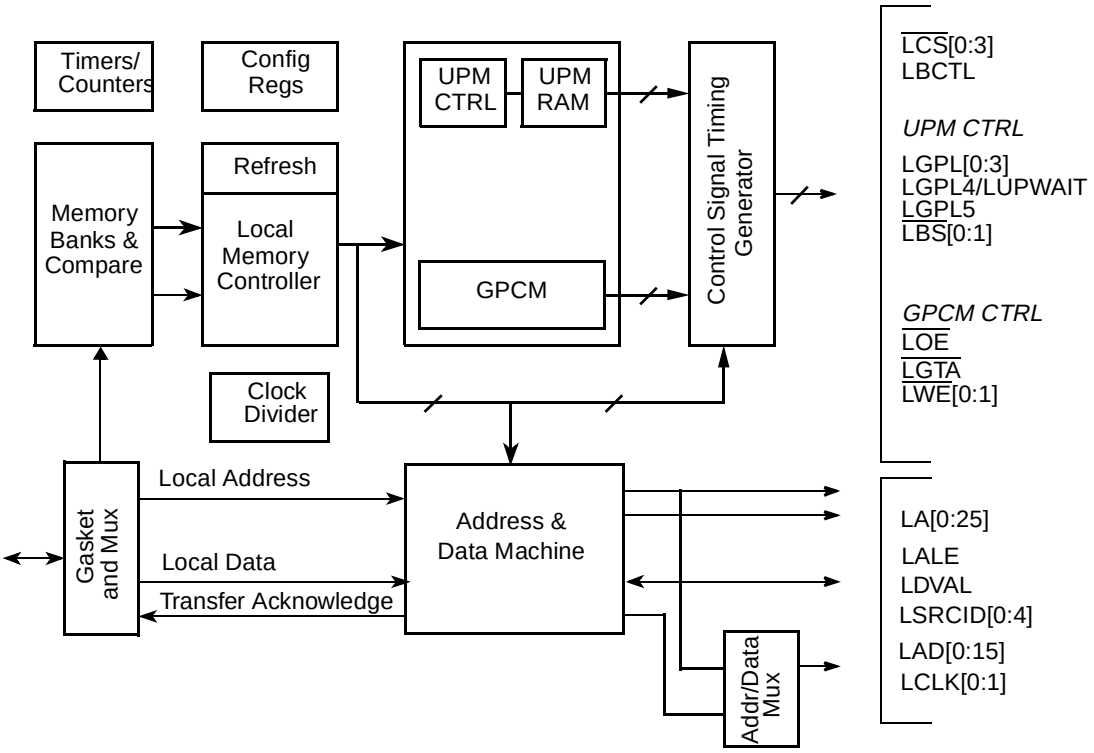


Figure 10-1. Local Bus Controller Block Diagram

The main component of the LBC is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling four memory banks shared by a GPCM and up to three UPMs. As such, it supports a minimal glue logic interface to 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 write protection and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

10.1.1 Features

The LBC main features are as follows:

- Memory controller with four memory banks
  - 32-bit address decoding with mask
  - Variable memory block sizes (32 Kbytes to 64 Mbytes)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - Automatic segmentation of large transactions
  - Write-protection capability
  - Atomic operation
- 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- and 16-bit devices
  - Minimum three-clock access to external devices
  - Two byte-write-enable signals ($\overline{\text{LWE}}$[0:1])
  - 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- and 16-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)

10.1.2 Modes of Operation

The LBC provides one GPCM and three UPMs for the local bus, with no restriction on how many of the four 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 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 or UPM mode, only one of the four chip selects is active at any time for the duration of the transaction. See Section 10.4, “Functional Description.”

10.1.2.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 2, 4, and 8 between the faster internal (system) clock and slower external bus clock (LCLK[0:1]). This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]). This ratio 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[0:1] to allow the clock load to be shared equally across a pair of signal nets, thereby enhancing the edge rates of the bus clock.

10.1.2.2 Source ID Debug Mode

The LBC provides the ID of a transaction source on external device signals. When those signals are selected, the 5-bit internal ID of the current transaction source appears on LSRCID[0:4] whenever valid address or data is available on the LBC external signals. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID signals at all other times. The combination of a valid source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (LDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on LSRCID[0:4] and LALE is asserted, a valid full 26-bit address may be latched from LAD[0:15] and combined with LA[16:25].
- If a valid source ID is detected on LSRCID[0:4] and LDVAL is asserted, valid data may be latched from LAD[0:15].

The LSRCID[0:4] and LDVAL signals are multiplexed with other functions sharing the same external signals. Refer to chapter 3, external signals description and to chapter 5, system configuration to learn how to enable the LSRCID/LDVAL signals.

10.2 External Signal Descriptions

Table 10-1 contains a list of external signals related to the LBC and summarizes their function. The table also shows the reset state of all external signals during assertion of $\overline{\text{HRESET}}$. For more information on the use of some of these signals as reset configuration signals on the MPC8323E, see Section 4.2.2, “Power-On Reset Flow.”

Table 10-1. Signal Properties—Summary

Name	Alternate Functions	Mode	Descriptions	No. of Signals	I/O	Reset State (Outputs)
LALE	—	—	External address latch enable	1	O	Reset_cfg
$\overline{\text{LCS0}}$	—	—	Chip select 0	1	O	Reset_cfg
$\overline{\text{LCS}}$[1:3]	—	—	Chip selects [1–3]	3	O	All high
$\overline{\text{LWE}}$[0:1]/ $\overline{\text{LBS}}$[0:1]	$\overline{\text{LWE}}$[0:1]	GPCM	Write enable	2	O	Reset_cfg
	$\overline{\text{LBS}}$[0:1]	UPM	Byte (lane) select			
LGPL0	LGPL0	UPM	General purpose line 0	1	O	Reset_cfg
LGPL1	LGPL1	UPM	General purpose line 1	1	O	Reset_cfg
$\overline{\text{LOE}}$/ LGPL2	$\overline{\text{LOE}}$	GPCM	Output enable	1	O	—
	LGPL2	UPM	General purpose line 2			
LGPL3	LGPL3	UPM	General purpose line 3	1	O	Reset_cfg
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT	$\overline{\text{LGTA}}$	GPCM	Transaction termination	1	I	High-Z
	LGPL4	UPM	General purpose line 4		O	
	LUPWAIT	UPM	External device wait		I	
LGPL5	—	UPM	General purpose line 5	1	O	Reset_cfg
LBCTL	—	—	Data buffer control	1	O	—
LA[0:25]	—	—	Local bus non-multiplexed address	26	O	—
LAD[0:15]	—	—	Multiplexed address/data bus	16	I/O	—
LCLK[0:1]	—	—	Local bus clocks	2	O	Driven
LDVAL	—	LBC debug	Local bus data valid	1	O	Not connected to external signals
LSRCID[0:4]	—	LBC debug	Local bus source ID	5	O	Not connected to external signals

Table 10-2 shows the detailed external signal descriptions for the LBC.

Table 10-2. Local Bus Controller Detailed Signal Descriptions

Signal	I/O	Description	
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device signals. See Section 10.4.1.2, "External Address Latch Enable Signal (LALE)."	
		State Meaning	Asserted/Negated—LALE is asserted with the address at the beginning of each memory controller transaction. The number of cycles for which it is asserted is governed by the ORn[EAD] and LCRR[EADC] fields. The exact timing of the negation of LALE is controlled by the RCWH[LALE] field. Note that no other control signals are asserted during the assertion of LALE.
$\overline{\text{LCS}}$[0:3]	O	Chip selects. Four chip selects are provided which are mutually exclusive.	
		State Meaning	Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. $\overline{\text{LCS}}$[0:3] 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.
$\overline{\text{LWE}}$[0:1]/ $\overline{\text{LBS}}$[0:1]	O	GPCM write enable/UPM byte select. These signals select or validate each byte lane of the data bus. For banks with port sizes of 16 bits (as set by BR*n*[PS]), both signals are defined. For an 8-bit port size, only bit 0 is defined. 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, $\overline{\text{LWE}}$[0:1] assert for each byte lane enabled for writing. $\overline{\text{LBS}}$[0:1] are programmable byte-select signals in UPM mode. See Section 10.4.3.4, "UPM RAM Array," for programming details about $\overline{\text{LBS}}$[0:1].
		Timing	Assertion/Negation—See Section 10.4.2, "General-Purpose Chip-Select Machine (GPCM)," for details regarding the timing of $\overline{\text{LWE}}$[0:1].
LGPL0	O	General purpose line 0	
		State Meaning	Asserted/Negated—One of six general purpose signals when in UPM mode; it drives a value programmed in the UPM array.
LGPL1	O	General-purpose line 1	
		State Meaning	Asserted/Negated—One of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LOE}}$/ LGPL2	O	GPCM output enable/General-purpose line 2	
		State Meaning	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. One of six general purpose lines when in UPM mode; it drives a value programmed in the UPM array.
LGPL3	O	General-purpose line 3	
		State Meaning	Asserted/Negated—This signal is one of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT	I/O	GPCM transfer acknowledge/General-purpose line 4/UPM wait	
		State Meaning	**Note:** 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. The $\overline{\text{LGTA}}$/LUPWAIT signal can be sampled as asserted (active-low) during LGPL4 is brought low by the UPM. For a subsequent GPCM transaction, it functions as $\overline{\text{LGTA}}$/LUPWAIT. As a result, GPCM transactions may be terminated prematurely before $\overline{\text{LGTA}}$/LUPWAIT has drifted to a logical one. **Work Around:** One way to resolve this issue is to ensure that $\overline{\text{LGTA}}$/LGPL4 is pulled-up to 3.3 V by an external 1-KΩ register. This will ensure that $\overline{\text{LGTA}}$ is sampled high (not asserted) by the time any subsequent GPCM transactions are initiated by the local bus memory controller. If this signal is used purely as an input ($\overline{\text{LGTA}}$/LUPWAIT), a weaker (10-KΩ) pull-up register may be substituted. A software work around to this errata is to program UPM. Therefore, it can pre-drive LGPL4 high prior to switching to input mode. A weak (10-KΩ or greater) pull-up is still required to maintain a stable level on $\overline{\text{LGTA}}$ for GPCM purposes.
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. Buffer control is disabled by setting OR*n*[BCTLD].	
		State Meaning	Asserted/Negated—Normally functions as a write/$\overline{\text{read}}$ control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.
LA[0:25]	O	Local bus nonmultiplexed address. All bits driven on LA[0:25] are defined for 8-bit port sizes. For 16-bit port sizes LA25 is a don't care.	
		State Meaning	Asserted/Negated—When the LBC is used in multiplexed address/data line configuration, LA[16:25] is used, unlatched, to connect to the ten least significant bits of the address for address phases. The other sixteen most significant bits of the address are taken from latched LAD[0:15]. For some RAM devices, such as fast-page DRAM, LA[21:25] serve as the column address offset during a burst access. When the LBC is used in non-multiplexed address/data line configuration, LA[0:25] is used, unlatched, to connect to the address line of the desired device.
LAD[0:15]	I/O	Multiplexed address/data bus. For configuration of a port size in BR*n*[PS] as 16 bits, LAD[0:7] must be connected to the most-significant byte lane (at address offset 0), while LAD[8:15] must be connected to the least-significant byte lane (at address offset 1). For a port size of 8 bits, only LAD[0:7] are connected to the external RAM.	
		State Meaning	Asserted/Negated—LAD[0:15] is the shared 16-bit address/data bus through which external RAM devices transfer data and receive addresses (16 most significant bits). The other ten least significant bits of the address are taken from the unlatched LA[16:25].
		Timing	Assertion/Negation—During assertion of LALE, LAD[0:15] are driven with the RAM address for the access to follow. External logic should propagate the address on LAD[0:15] while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD[0:15] 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[0:15] are again taken into a high-impedance state.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LCLK[0:1]	O	Local bus clocks. Enabled by the LCKENR register.	
		State Meaning	Asserted/Negated—LCLK[0:1] drive an identical bus clock signal for distributed loads.
LDVAL	O	Local bus data valid (LBC debug mode only)	
		State Meaning	Asserted/Negated—For a read, LDVAL asserts for one bus cycle in the cycle immediately preceding the sampling of read data on LAD[0:15]. For a write, LDVAL asserts for one bus cycle during the final cycle for which the current write data on LAD[0:15] is valid. During burst transfers, LDVAL asserts for each data beat.
		Timing	Assertion/Negation—Valid only while the LBC is in system debug mode. In debug mode, LDVAL asserts when the LBC generates a data transfer acknowledge.
LSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all LSRCID[0:4] signals are driven high unless LSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		State Meaning	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until LDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, LSRCID[0:4] is valid only when the address on LAD[0:15] consists of all physical address bits—with optional padding—for reconstructing the system address presented to the LBC.

10.3 Memory Map/Register Definition

Table 10-3 shows the memory-mapped registers of the LBC. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 10-3. Local Bus Controller Memory Map

Address Offset	Use	Access	Reset	Section/Page
0x0_5000	BR0—Base register 0	R/W	0x0000_RR01[1]	10.3.1.1/10-8
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020–0x0_5038	Reserved	—	—	—
0x0_5004	OR0—Option register 0	R/W	0x0000_0FF7	10.3.1.2/10-10
0x0_500C	OR1—Option register 1		0x0000_0000	
0x0_5014	OR2—Option register 2			
0x0_501C	OR3—Option register 3			
0x0_5024–0x0_503C	Reserved	—	—	—
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	10.3.1.3/10-14
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	10.3.1.4/10-15

Table 10-3. Local Bus Controller Memory Map (continued)

Address Offset	Use	Access	Reset	Section/Page
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	10.3.1.4/10-15
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	10.3.1.4/10-15
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	10.3.1.5/10-17
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	10.3.1.6/10-18
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	10.3.1.7/10-18
0x0_50B0	LTESR—Transfer error status register	Read/bit-reset	0x0000_0000	10.3.1.8/10-19
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	10.3.1.9/10-20
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	10.3.1.10/10-21
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	10.3.1.11/10-22
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	10.3.1.12/10-23
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	10.3.1.13/10-23
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	10.3.1.14/10-24

[1] Port size for BR0 is configured from the value of RCWH[ROMLOC], which is loaded during reset, hence ‘RR’ is either 0x08 or 0x10.

10.3.1 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 10-3 should not be accessed for reading or writing. Similarly, only zero should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

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

10.3.1.1 Base Registers (BR0–BR3)

The base registers (BR*n*), shown in Figure 10-2, 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]–BR3[V] are cleared, and the value of BR0[PS] reflects the initial port size configured by the boot ROM location field of the reset configuration word.

Offset 0x0_5000 (BR0) Access: Read/Write
0x0_5008 (BR1)
0x0_5010 (BR2)
0x0_5018 (BR3)

	0															16	17	18	19	20	21	22	23	24		26	27	28		29	30	31
R W	BA																—		PS		—		WP	MSEL			—		ATOM		—	V
Reset[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P	S	0	0	0	0	0	0	0	0	0	0	0/1

[1] BR0 has its valid bit set during reset. Thus bank 0 is valid with the port size (PS) configured from RCWH[ROMLOC] as loaded during reset. All other base registers have all bits cleared to zero during reset.

Figure 10-2. Base Registers (BR*n*)

Table 10-4 describes BR*n* fields.

Table 10-4. 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 field in reset configuration word as loaded 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 Reserved
21–22	—	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 Reserved 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27	—	Reserved

Table 10-4. BR*n* Field Descriptions (continued)

Bits	Name	Description
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.

10.3.1.2 Option Registers (OR0–OR3)

The OR*n* registers define the sizes of memory banks and access attributes. The OR*n* attribute bits support the following two modes of operation as defined by BR*n*[MSEL].

- GPCM mode (see Section 10.3.1.2.2, “Option Registers (ORn)—GPCM Mode”)
- UPM mode (see Section 10.3.1.2.3, “Option Registers (ORn)—UPM Mode”)

The OR*n* registers are interpreted differently depending on which of the three machine types is selected for that bank.

10.3.1.2.1 Address Mask

The address mask fields of the option registers (OR*n*[AM]) mask up to 17 bits of the corresponding BR*n*[BA] fields. The 15 lsbs of the 32-bit internal address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. Table 10-5 shows memory bank sizes from 256 Kbytes to 64 Mbytes.

Table 10-5. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes

Table 10-5. Memory Bank Sizes in Relation to Address Mask (continued)

AM	Memory Bank Size
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

10.3.1.2.2 Option Registers (OR*n*)—GPCM Mode

Figure 10-3 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the GPCM machine.

Offset 0x0_5004 (OR0)
0x0_500C (OR1)
0x0_5014 (OR2)
0x0_501C (OR3)

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 [1]	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
Reset OR1–OR7	All zeros											

[1] OR0 has this value set during reset (GPCM is the default control machine for all banks coming out of reset). All other option registers have all bits cleared.

Figure 10-3. Option Registers (OR*n*) in GPCM Mode

Table 10-6 describes OR*n* fields for GPCM mode.

Table 10-6. OR*n*—GPCM Field Descriptions

Bits	Name	Description
0–16	AM	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. See Section 10.3.1.2.1, “Address Mask.”
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.

Table 10-6. OR*n*—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>20</td><td>CSNT</td><td>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 earlier depending on the value of LCRR[CLKDIV].
<table>
<tr><th>LCRR[CLKDIV]</th><th>CSNT</th><th>Meaning</th></tr>
<tr><td>x</td><td>0</td><td>$\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally.</td></tr>
<tr><td>2</td><td>1</td><td>$\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally.</td></tr>
<tr><td>4 or 8</td><td>1</td><td>$\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated quarter of a bus clock cycle earlier.</td></tr>
</table></td></tr>
<tr><td>21–22</td><td>ACS</td><td>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
<table>
<tr><th>LCRR[CLKDIV]</th><th>Value</th><th>Meaning</th></tr>
<tr><td>x</td><td>00</td><td>$\overline{\text{LCS}n}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.</td></tr>
<tr><td></td><td>01</td><td>Reserved.</td></tr>
<tr><td>2</td><td>10</td><td>$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.</td></tr>
<tr><td></td><td>11</td><td>$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.</td></tr>
<tr><td>4 or 8</td><td>10</td><td>$\overline{\text{LCS}n}$ is output a quarter bus clock cycle after the address lines.</td></tr>
<tr><td></td><td>11</td><td>$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.</td></tr>
</table></td></tr>
<tr><td>23</td><td>XACS</td><td>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 10-20 and Table 10-21 for LCRR[CLKDIV] = 4 or 8, Table 10-22 and Table 10-23 for LCRR[CLKDIV] = 2).</td></tr>
<tr><td>24–27</td><td>SCY</td><td>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</td></tr>
<tr><td>28</td><td>SETA</td><td>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).</td></tr>
</table>

Table 10-6. OR*n*—GPCM Field Descriptions (continued)

Bits	Name	Description
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 is not equal to 00). • Doubles the number of wait states specified by SCY, providing up to 30 wait states. • Works in conjunction with EHTR to extend hold time on read accesses. • $\overline{\text{LCS}n}$ (only if ACS is not equal to 00) and $\overline{\text{LWE}}$ signals are negated one cycle earlier during writes.
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access. TRLX / EHTR / Meaning: 0 / 0 — The memory controller generates normal timing. No additional cycles are inserted. 0 / 1 — 1 idle clock cycle is inserted. 1 / 0 — 4 idle clock cycles are inserted. 1 / 1 — 8 idle clock cycles are inserted.
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

10.3.1.2.3 Option Registers (OR*n*)—UPM Mode

Figure 10-4 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects a UPM machine.

Offset 0x0_5004 (OR0) Access: Read/Write
0x0_500C (OR1)
0x0_5014 (OR2)
0x0_501C (OR3)

	0				16	17	18	19	20	22	23	24	28	29	30	31
R W	AM					—		BCTLD	—		BI	—		TRLX	EHTR	EAD
Reset	All zeros															

Figure 10-4. Option Registers (OR*n*) in UPM Mode

Table 10-7 describes OR*n* fields for UPM mode.

Table 10-7. OR*n*—UPM Field Descriptions

Bits	Name	Description
0–16	AM	UPM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	—	Reserved
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20–22	—	Reserved
23	BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.
24–28	—	Reserved
29	TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.
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.
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]).

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.

10.3.1.3 UPM Memory Address Register (MAR)

Figure 10-5 shows the fields of the UPM memory address register (MAR).

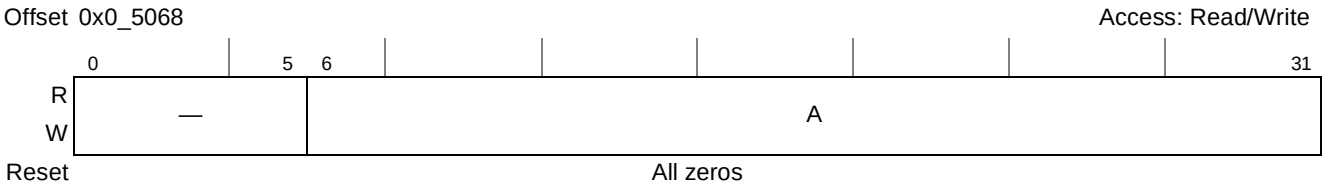


Figure 10-5. UPM Memory Address Register (MAR)

Table 10-8 describes the MAR fields.

Table 10-8. MAR Field Descriptions

Bits	Name	Description
0–5	—	Reserved, must be set to zero.
6–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

10.3.1.4 UPM Mode Registers (M*n*MR)

The UPM machine mode registers (MAMR, MBMR, and MCMR), shown in Figure 10-6, contain the configuration for the three UPMs.

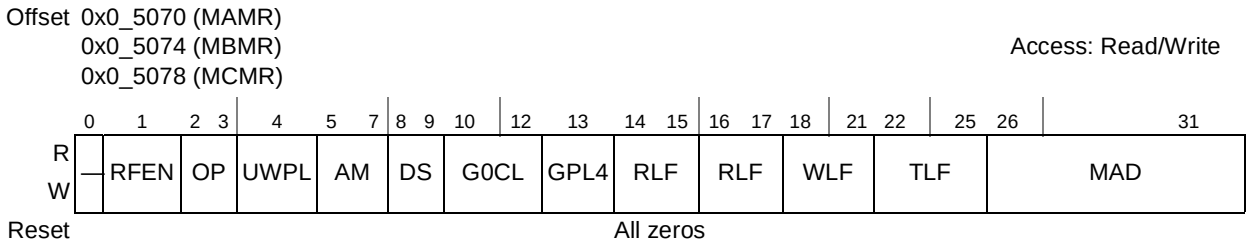


Figure 10-6. UPM Mode Registers (M*n*MR)

Table 10-9 describes UPM mode fields.

Table 10-9. M*n*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. 00 Normal operation 01 Write to UPM array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed to by MAD. After the access, MAD is automatically incremented. 10 Read from UPM array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed to by MAD into the MDR. After the access, MAD is automatically incremented. 11 Run pattern. On the next memory access that hits a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM word.
4	UWPL	LUPWAIT polarity active low. Sets the polarity of the LUPWAIT signal when in UPM mode. 0 LUPWAIT is active high. 1 LUPWAIT is active low.

Table 10-9. M*n*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–LA9</th><th>LA10</th><th>LA11</th><th>LA12</th><th>LA13–LA22</th><th>LA23</th><th>LA24</th><th>LA25</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 10-26 on page 10-49.
<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 10-9. M*n*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* is executed for a burst- or single-beat read pattern or when M*n*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* is executed for a burst- or single-beat write pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
22–25	TLF	Refresh loop field. Determines the number of times a loop defined in the UPM*n* is executed for a refresh service pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
26–31	MAD	Machine address. RAM address pointer for the command executed. This field is incremented by 1, each time the UPM is accessed and the OP field is set to WRITE or READ. Address range is 64 words per UPM*n*.

10.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

MRTPR shown in Figure 10-7, is used to divide the system clock to provide the UPM refresh timers clock.

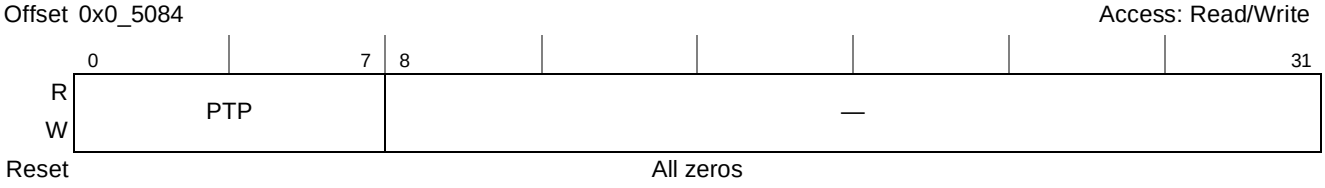


Figure 10-7. Memory Refresh Timer Prescaler Register (MRTPR)

Table 10-10 describes MRTPR fields.

Table 10-10. MRTPR Field Descriptions

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The system clock is divided by PTP except when the value is 0x0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

10.3.1.6 UPM Data Register (MDR)

MDR shown in Figure 10-8, contains data written to or read from the RAM array for UPM read or write commands. MDR must be set up before issuing a write command to the UPM.

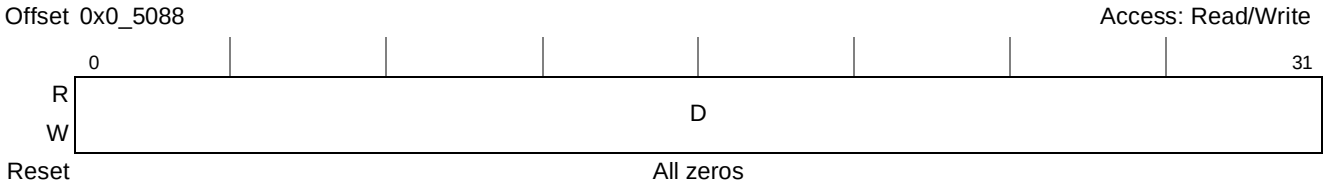


Figure 10-8. UPM Data Register (MDR)

Table 10-11 describes MDR[D].

Table 10-11. MDR Field Description

Bits	Name	Description
0–31	D	The data to be read or written into the RAM array when a write or read command is supplied to the UPM (M*n*MR[OP] = 01 or M*n*MR[OP] = 10).

10.3.1.7 UPM Refresh Timer (LURT)

LURT, shown in Figure 10-9, generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled (M*n*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.

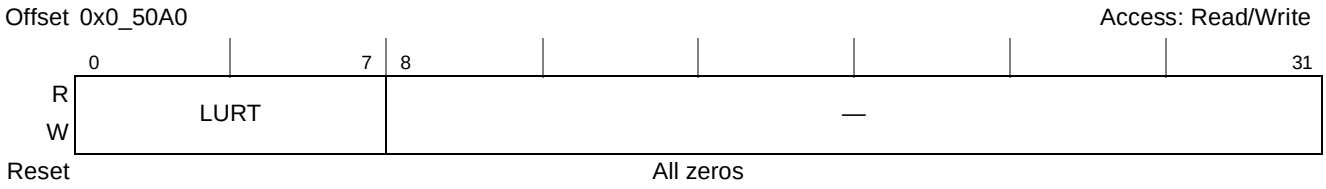


Figure 10-9. UPM Refresh Timer (LURT)

Table 10-12 describes LURT fields.

Table 10-12. 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: $TimerPeriod = \frac{LURT}{\left(\frac{Fsystemclock}{MRTPR[PTP]}\right)}$ 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 x MRTPR[PTP] system clock cycles.
8–31	—	Reserved

10.3.1.8 Transfer Error Status Register (LTESR)

The LBC has five 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 10-10, is a write-1-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, 0b0400_0000 should be written to the register.

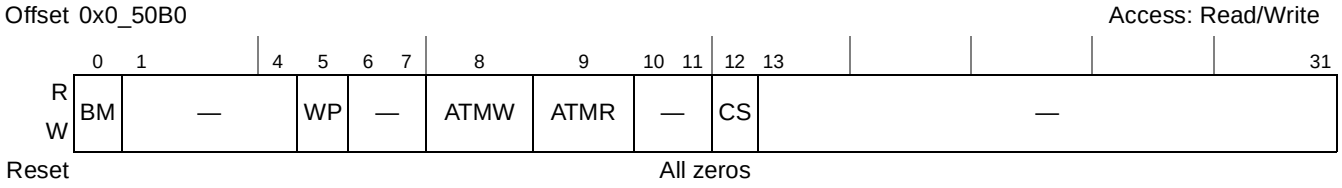


Figure 10-10. Transfer Error Status Register (LTESR)

Table 10-13 describes LTESR fields.

Table 10-13. 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] x 8 bus clock cycles from the start of a transaction.
1–4	—	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 write0No atomic write error occurred. 1 The subsequent write (WARA) to a memory bank did not occur within 256 bus clock cycles.
9	ATMR	Atomic error read0No 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

10.3.1.9 Transfer Error Check Disable Register (LTEDR)

LTEDR shown in Figure 10-11, 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.

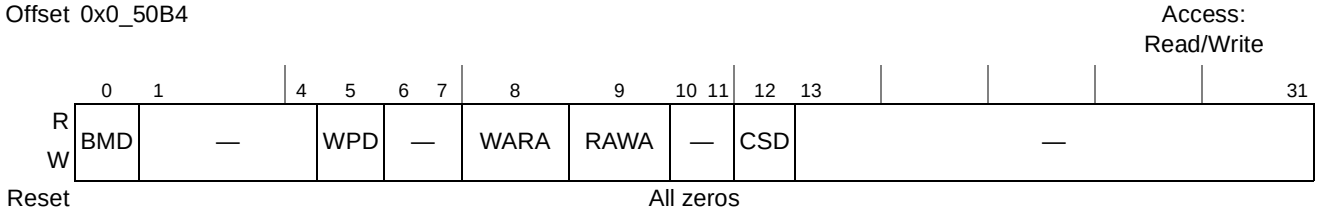


Figure 10-11. Transfer Error Check Disable Register (LTEDR)

Table 10-14 describes LTEDR fields.

Table 10-14. LTEDR Field Descriptions

Bits	Name	Description
0	BMD	Bus monitor disable. 0 Bus monitor is enabled. 1 Bus monitor is disabled.
1–4	—	Reserved

Table 10-14. LTEDR Field Descriptions (continued)

Bits	Name	Description
5	WPD	Write protect error checking disable. 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–7	—	Reserved
8	WARA	Write after read atomic (WARA) error checking disable. 0 WARA error checking is enabled. 1 WARA error checking is disabled.
9	RAWA	Read after write atomic (RAWA) error checking disable. 0 RAWA error checking is enabled. 1 RAWA error checking is disabled.
10–11	—	Reserved
12	CSD	Chip select error checking disable. 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–31	—	Reserved

10.3.1.10 Transfer Error Interrupt Enable Register (LTEIR)

LTEIR shown in Figure 10-12, 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.

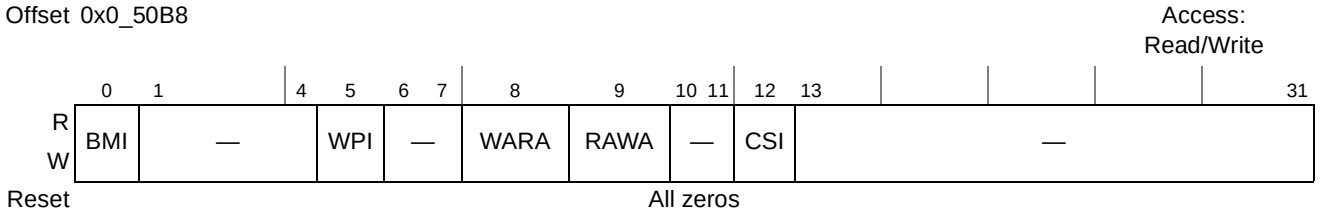


Figure 10-12. Transfer Error Interrupt Enable Register (LTEIR)

Table 10-15 describes LTEIR fields.

Table 10-15. 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.
1–4	—	Reserved
5	WPI	Write protect error interrupt enable. 0 Write protect error reporting is disabled. 1 Write protect error reporting is enabled.
6–7	—	Reserved

Table 10-15. LTEIR Field Descriptions (continued)

Bits	Name	Description
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

10.3.1.11 Transfer Error Attributes Register (LTEATR)

LTEATR is shown in Figure 10-13. After LTEATR[V] has been set, software must clear this bit to allow LTEATR and LTEAR to update following any subsequent errors.

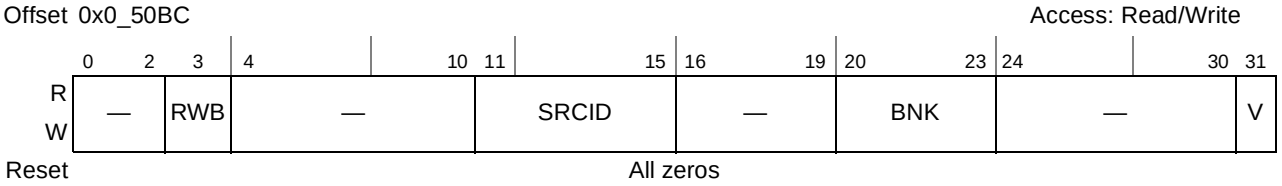


Figure 10-13. Transfer Error Attributes Register (LTEATR)

Table 10-16 describes LTEATR fields.

Table 10-16. LTEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	RWB	Transaction type for the error: 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.
4–10	—	Reserved
11–15	SRCID	Captures the source of the transaction when this information is provided on the internal interface to the LBC. The coding of the source ID debug information is the same as the coding of AEATR[MSTR_ID] (see Section 6.2.6, “Arbiter Event Attributes Register (AEATR).”)
16–19	—	Reserved
20–23	BNK	Memory controller bank. There is one error status bit per memory controller bank (bit 20 represents bank 0). A bit is set for the local bus memory controller bank that had an error.

Table 10-16. LTEATR Field Descriptions (continued)

Bits	Name	Description
24–30	—	Reserved
31	V	Error attribute capture is valid. Indicates that the captured error information is valid. 0 Captured error attributes and address are not valid. 1 Captured error attributes and address are valid.

10.3.1.12 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) is shown in Figure 10-14.

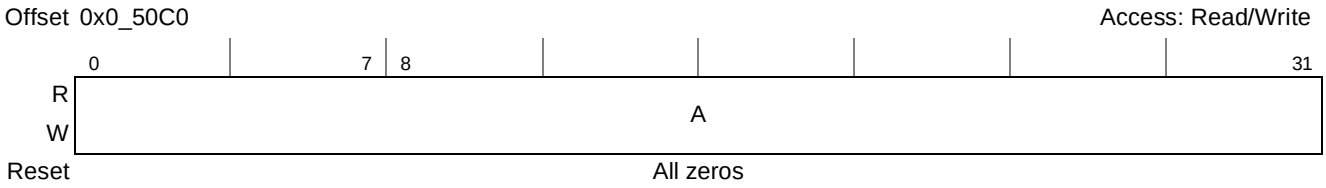


Figure 10-14. Transfer Error Address Register (LTEAR)

Table 10-17 describes LTEAR[A].

Table 10-17. 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.

10.3.1.13 Local Bus Configuration Register (LBCR)

LBCR is shown in Figure 10-15.

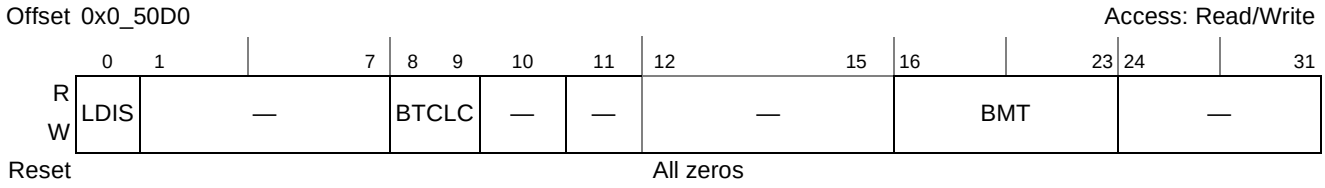


Figure 10-15. Local Bus Configuration Register

Table 10-18 describes LBCR fields.

Table 10-18. 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

Table 10-18. LBCR Field Descriptions (continued)

Bits	Name	Description
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.
10–15	—	Reserved
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.
24–31	—	Reserved

10.3.1.14 Clock Ratio Register (LCRR)

The clock ratio register, shown in, sets the system clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

NOTE

For proper operation of the system, this register setting must not be altered while local bus memories or devices are being accessed. Special care needs to be taken when running instructions from an local bus controller memory.

Offset 0x0_50D4 Access: Read/Write

	0	1	2	4	5	6	7	8	13	14	15	16	27	28	31
R	—									EADC		—		CLKDIV	
W															
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 10-16. Clock Ratio Register (LCRR)

Table 10-19 describes LCRR fields.

Table 10-19. LCRR Field Descriptions

Bits	Name	Description
0–13	—	Reserved
14–15	EADC	External address delay cycles.Defines the number of cycles for the assertion of LALE. 00 4 01 1 10 2 11 3

Table 10-19. LCRR Field Descriptions (continued)

Bits	Name	Description
16–27	—	Reserved
28–31	CLKDIV	System (input) clock divider. Sets the frequency ratio between the (input) system clock and the memory bus clock. Only the values shown in the table below are allowed. 0000–0001 reserved 0010 2 0011 reserved 0100 4 0101–0111 reserved 1000 8 1001–1111 reserved

10.4 Functional Description

The LBC allows the implementation of memory systems with very specific timing requirements.

- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. It has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals.
- The UPM supports refresh timers, address multiplexing of the external bus and generation of programmable control signals for row address and column address strobes, to allow for a minimal glue logic interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.

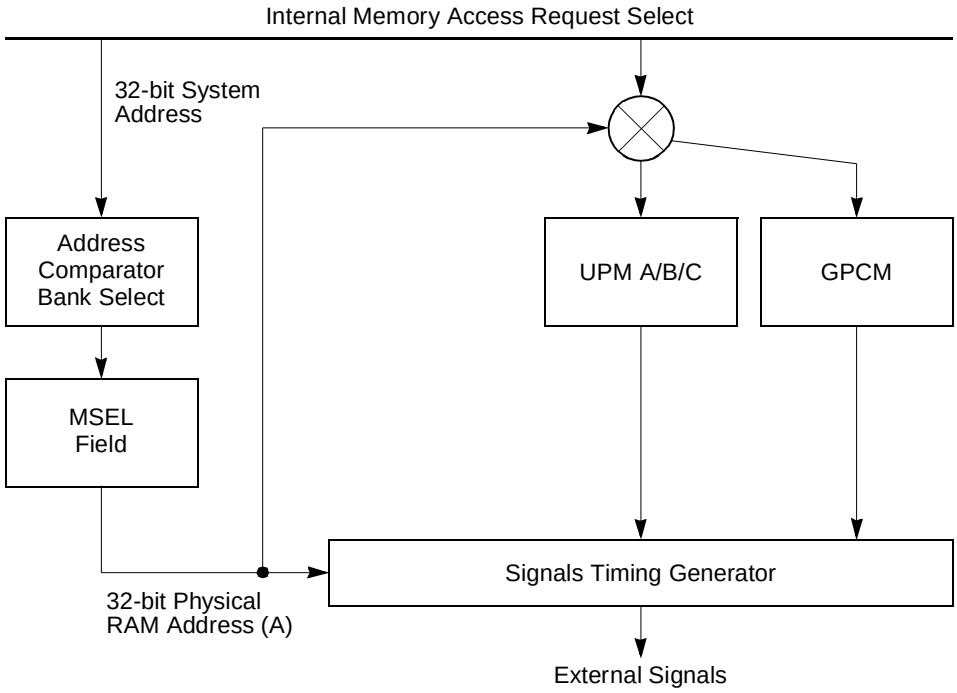


Figure 10-17. Basic Operation of Memory Controllers in the LBC

Each memory bank (chip select) can be assigned to either of these two type of machines through the machine select bits of the base register for that bank (BR*n*[MSEL]), as illustrated in Figure 10-17. If a bank match occurs, the corresponding machine (GPCM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

10.4.1 Basic Architecture

These subsections describe the basic architecture of the LBC.

10.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BR*n* registers; the corresponding address masks are written to the OR*n* registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 17 msbs of the address, masked by OR*n*[AM], with the base address for each bank (BR*n*[BA]). If a match is found on a memory controller bank, the attributes defined in the BR*n* and OR*n* for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

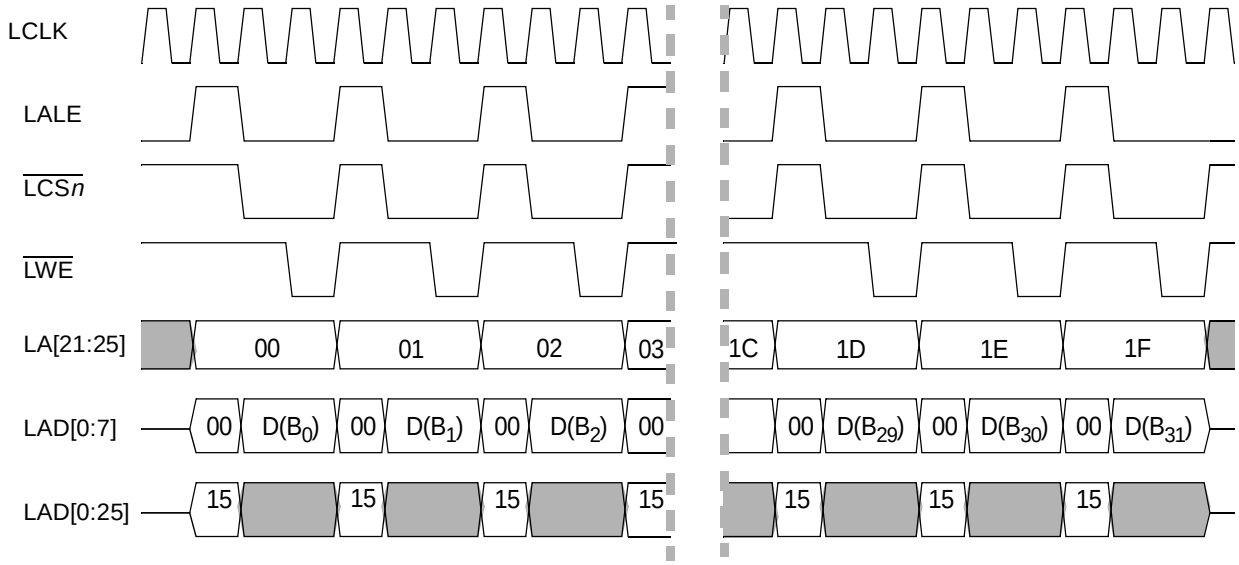
10.4.1.2 External Address Latch Enable Signal (LALE)

The local bus uses a multiplexed address/data bus. Therefore the LBC must distinguish between address (the 16-MSB) and data phases, which take place on the same bus (LAD[0:15] signals). The LALE signal, when asserted, signifies an address phase during which the LBC drives the 16-MSB memory address on

the LAD[0:15] 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. The other 10-LSB address is provided by LA[16:25], unlatched, to the address signals of the memory or peripheral device. When LALE is negated, LAD[0:15] then serves as the (bi-directional) data bus for the access. Any address phase initiates the assertion of LALE, which has a programmable duration of between 1 and 4 bus clock cycles.

The frequency of LALE assertion varies across the two 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. In the case of UPM, the frequency of LALE assertion depends on how the UPM RAM is programmed. UPM single accesses typically assert LALE once, upon commencement, but it is possible to program UPM to assert LALE several times, and to change the values of LA[21:25] with and without LALE being involved. In general, when using the GPCM controller it is not necessary to use LA[21:25] if a sufficiently wide latch is used to capture the entire address during LALE phases. UPM may require LA[21:25] if the LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the LBC, Figure 10-18 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[21:25] exactly mirror LAD[21:25], but during data phases, only LAD[0:7] is driven with valid data.



Note: All address and signal values are shown in hexadecimal.
D(Bk) = kth of 32 data bytes.

Figure 10-18. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420

10.4.1.3 Data Transfer Acknowledge (TA)

The two memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:15] to be either sampled (for reads) or changed (on writes). The data sampling/data change always occurs at the end of the bus cycle in which the LBC asserts TA internally. In LBC debug mode, TA is also visible externally on the LDVAL signal. GPCM controller automatically generates TA according to the timing parameters programmed for them in option and mode registers; a UPM generates TA only when

a UPM pattern has the UTA RAM word bit set. Figure 10-19 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.

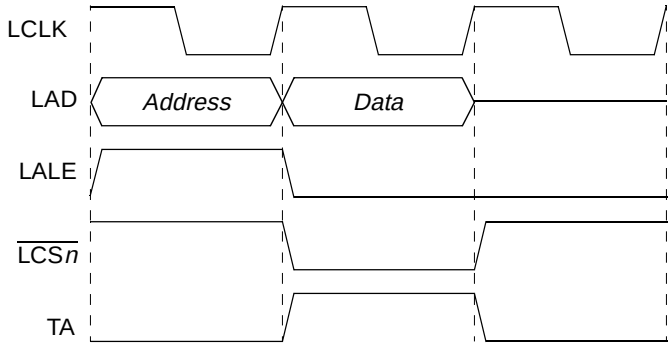


Figure 10-19. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}n}$

10.4.1.4 Data Buffer Control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL). This signal is activated when a GPCM or UPM controlled bank is accessed. LBCTL can be disabled by setting OR*n*[BCTLD]. 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.

10.4.1.5 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).

10.4.1.6 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 (such as the LTESR[BM] bit is not set by a bus monitor time-out); however, the bus monitor is still active and can generate a UPM exception (as noted in Section 10.4.3.1.4, “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— resulting in incomplete data transfers. Accordingly, apart from the reset value of 0x00 (corresponding with the maximum time-out of 2048 bus cycles), LBCR[BMT] must not be set below 0x05 (or 40 bus cycles for time-out) under any circumstances.

10.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPROM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups—BR*n* and OR*n*.

Figure 10-20 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 ($\overline{\text{LCS0}}$) before to the system is fully configured.

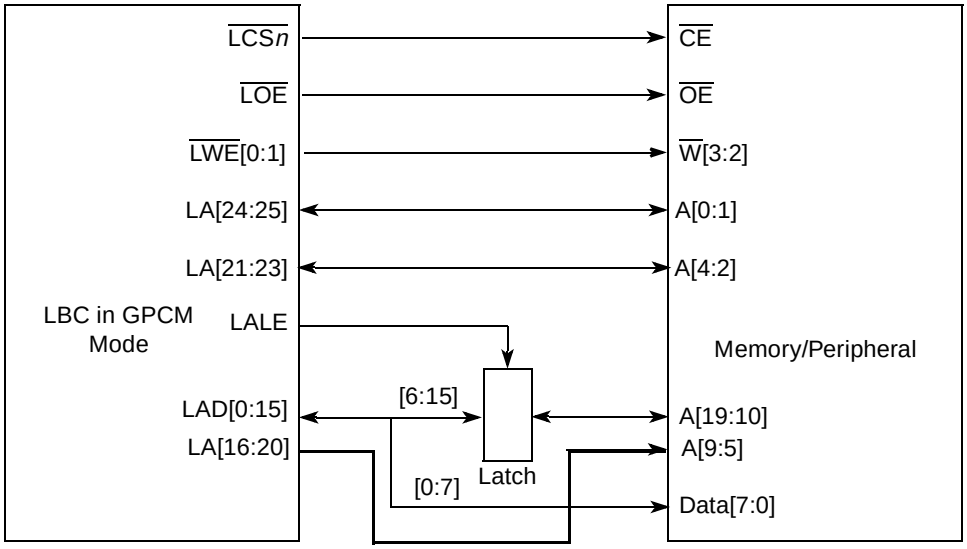


Figure 10-20. Local Bus to GPCM Device Interface

Figure 10-21 shows $\overline{\text{LCS}}$ as defined by the setup time required between the address lines and $\overline{\text{CE}}$. The user can configure OR*n*[ACS] to specify $\overline{\text{LCS}}$ to meet this requirement.

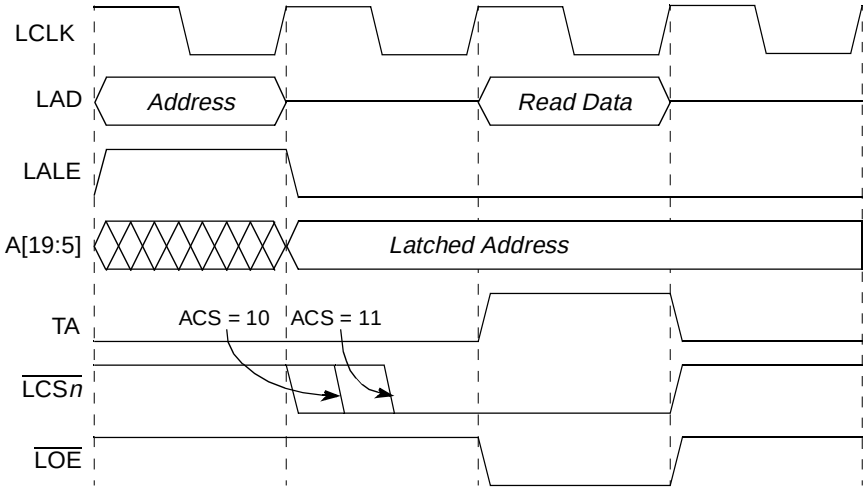


Figure 10-21. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)

10.4.2.1 Timing Configuration

If BR*n*[MSEL] selects the GPCM, the attributes for the memory cycle are taken from OR*n*. These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR and SETA fields. Table 10-20 shows signal behavior and system response for a write access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 10-21 shows the signal behavior and system response for a read access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 10-22 and Table 10-23 show the write and read signal behavior, respectively, when LCRR[CLKDIV] = 2.

Table 10-20. 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
1	0	00	0	0	0	0	3+2*SCY
1	0	10	0	1+1/4	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	–1–1/4	4+2*SCY
1	0	10	1	1+1/4	–1–1/4	–1–1/4	5+2*SCY
1	0	11	1	1+1/2	–1–1/4	–1–1/4	5+2*SCY
1	1	00	1	0	0	–1–1/4	4+2*SCY
1	1	10	1	2	–1–1/4	–1–1/4	5+2*SCY
1	1	11	1	3	–1–1/4	–1–1/4	6+2*SCY

[1] Total cycles when LALE is asserted for one cycle only (OR*n*[EAD] = 0; OR*n*[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 10-21. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (bus clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ Negated to Address Change	Total Cycles[1]
0	0	0	00	0	1	4+SCY
0	0	0	10	1/4	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/4	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/4	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/4	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

1 Total cycles when LALE is asserted for one cycle only (OR*n*[EAD] = 0; OR*n*[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 10-22. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCSn}}$ Asserted	$\overline{\text{LCS}n}$ 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
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 10-23. 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
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/2	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

1 Total cycles when LALE is asserted for one cycle only (OR*n*[EAD]=0; OR*n*[EAD]=1 and LCRR[EADC]=01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

10.4.2.2 Chip-Select Assertion Timing

The banks selected to work with the GPCM support an option to drive the $\overline{\text{LCS}n}$ 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 and not the address timing on LAD[0:15]. That is, the chip select does not assert during LALE).

- One quarter of a clock cycle later (for LCRR[CLKDIV] = 4, 8).
- One half of a clock cycle later (for LCRR[CLKDIV] = 2, 4, or 8).
- One clock cycle later (for LCRR[CLKDIV] = 4), when OR*n*[XACS] = 1.
- Two clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR*n*[XACS] = 1.
- Three clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR*n*[XACS] = 1 and OR*n*[TRLX] = 1.

Figure 10-21 shows two chip-select assertion timings for the case LCRR[CLKDIV] = 4 or 8. If LCRR[CLKDIV] = 2, $\overline{\text{LCS}n}$ asserts identically for OR*n*[ACS] = 10 or 11.

10.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between 0 and 30 wait states to be added to an access by programming OR*n*[SCY] and OR*n*[TRLX]. Internal generation of transfer acknowledge is enabled if OR*n*[SETA] = 0. If $\overline{\text{LGTA}}$ is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by $\overline{\text{LGTA}}$; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of OR*n*[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}n}$ in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from OR*n*[SCY] cycles to 2*OR*n*[SCY] cycles, allowing a maximum of 30 wait states.

10.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 10-20 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}n}$ is connected directly to $\overline{\text{CE}}$ of the memory device. $\overline{\text{LWE}}$[0:1] are connected to the respective $\overline{\text{WE}}$[1:0] signals on the memory device where each $\overline{\text{LWE}}$[0:1] signal corresponds to a different data byte.

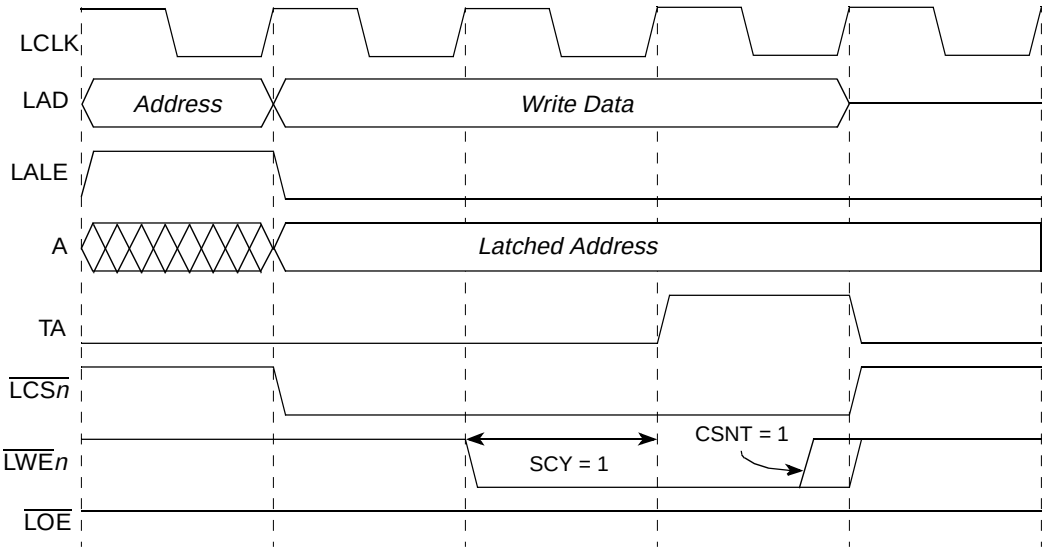


Figure 10-22. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4, 8)

As Figure 10-22 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 the figure). OR*n*[CSNT] controls the timing for the appropriate strobe negation in write cycles. When this attribute is asserted, the strobe is negated one quarter of a clock before the normal case provided that LCRR[CLDIV] = 4 or 8. For example, when ACS = 00 and CSNT = 1, $\overline{\text{LWE}}n$ is negated one quarter of a clock earlier, as shown in Figure 10-22. If LCRR[CLDIV] = 2, $\overline{\text{LWE}}n$ is negated either coincident with $\overline{\text{LCS}}n$ or one cycle earlier.

10.4.2.2.3 Relaxed Timing

OR*n*[TRLX] is provided for memory systems that require more relaxed timing between signals. Setting TRLX = 1 has the following effect on timing:

- An additional bus cycle is added between the address and control signals (but only if ACS is not equal to 00).
- The number of wait states specified by SCY is doubled, providing up to 30 wait states.
- The extended hold time on read accesses (EHTR) is extended further.
- $\overline{\text{LCS}}n$ signals are negated 1-cycle earlier during writes (if ACS ≠00).
- $\overline{\text{LWE}}$[0:1] signals are negated one cycle earlier during writes.

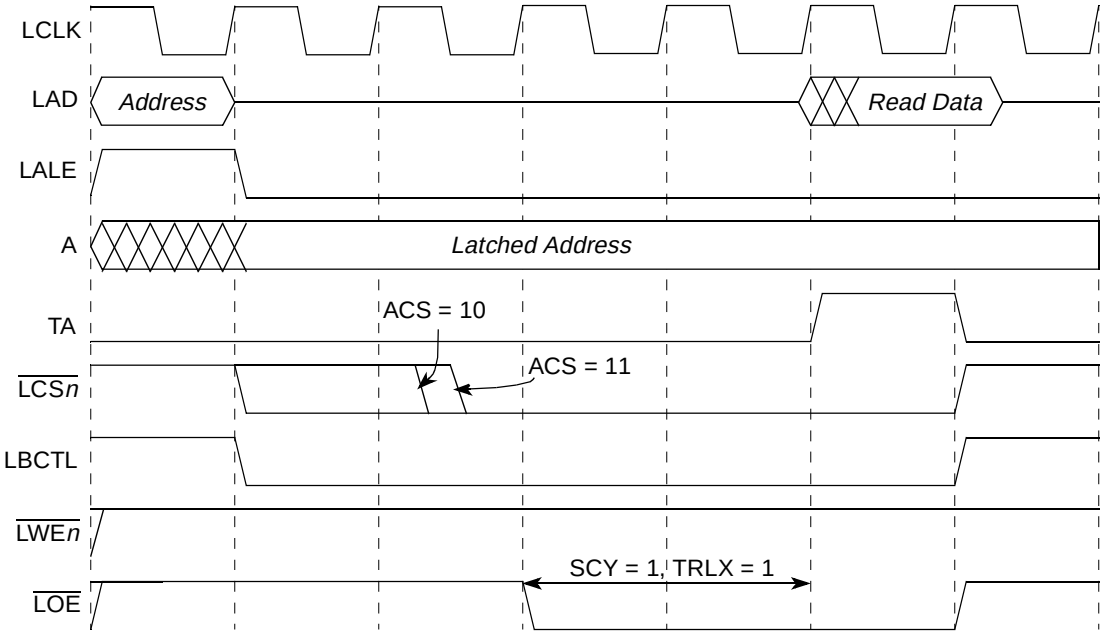


Figure 10-23. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

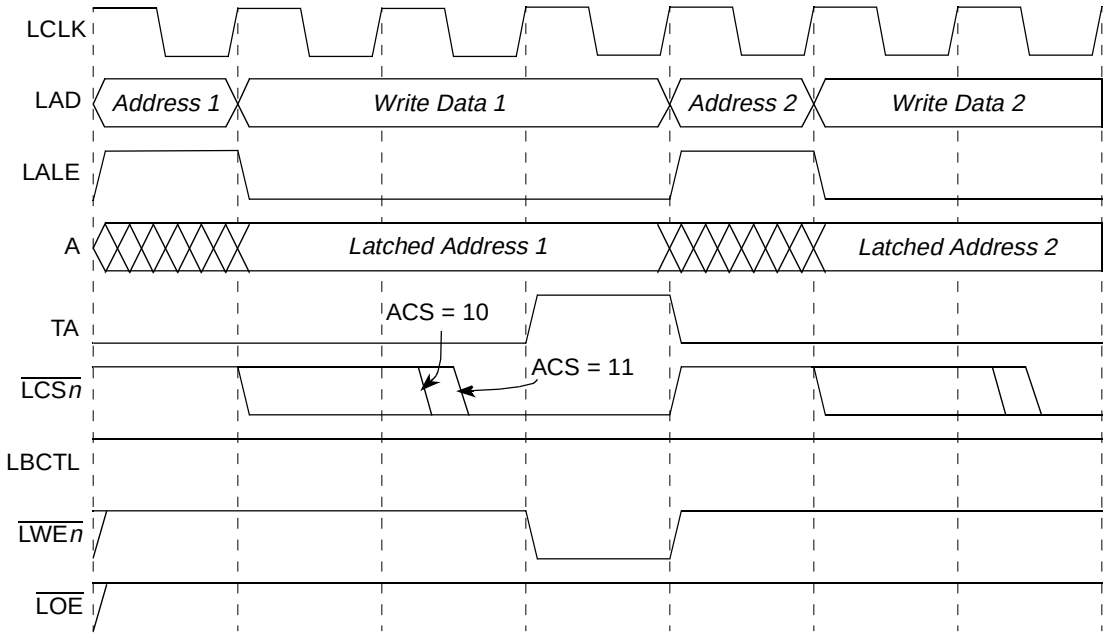


Figure 10-24. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

Figure 10-23 and Figure 10-24 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 10-24 also shows address and data multiplexing on LAD[0:15] for a pair of writes issued consecutively.

When TRLX and CSNT are set in a write access, the $\overline{\text{LWE}}$[0:1] strobe signals are negated one clock earlier than in the normal case, as shown in Figure 10-25 and Figure 10-26. If ACS ≠ 00, $\overline{\text{LCS}n}$ is also negated one clock earlier.

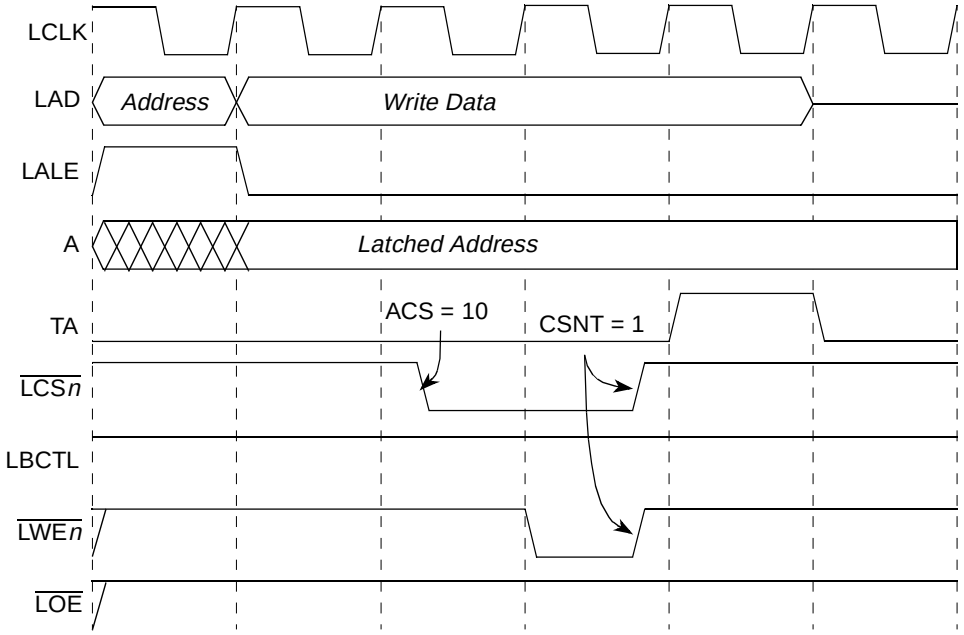


Figure 10-25. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

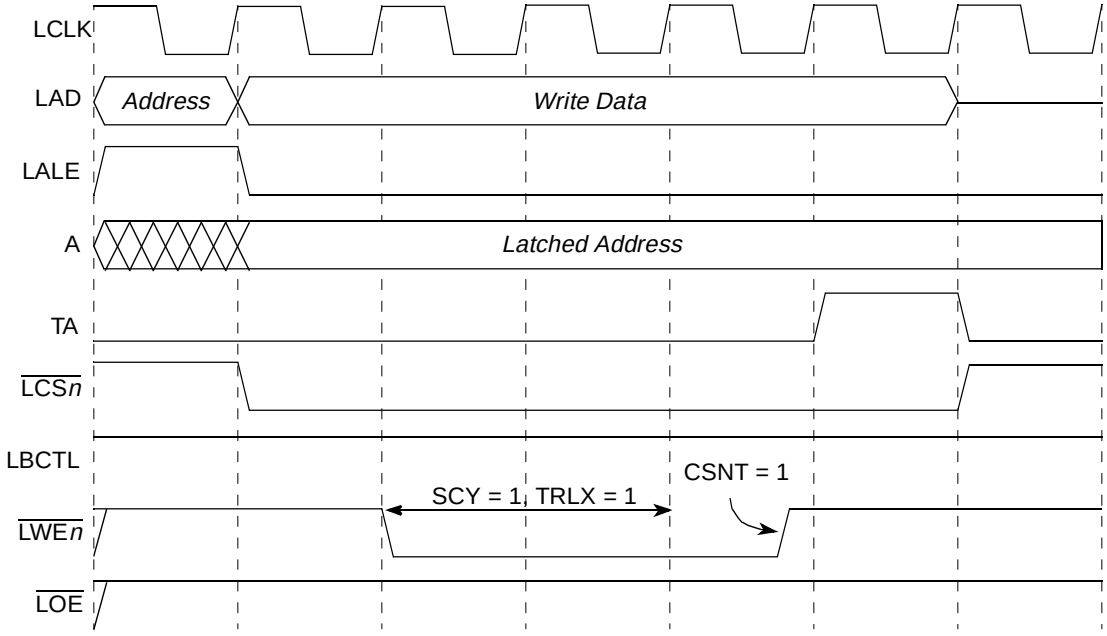


Figure 10-26. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

10.4.2.2.4 Output Enable ($\overline{\text{LOE}}$) Timing

The timing of the $\overline{\text{LOE}}$ 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 ACS = 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

10.4.2.2.5 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of OR*n*[TRLX,EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in Table 10-6 in addition to any existing bus turnaround cycle. The final bus turnaround cycle is automatically inserted by the LBC for reads, regardless of the setting of OR*n*[EHTR].

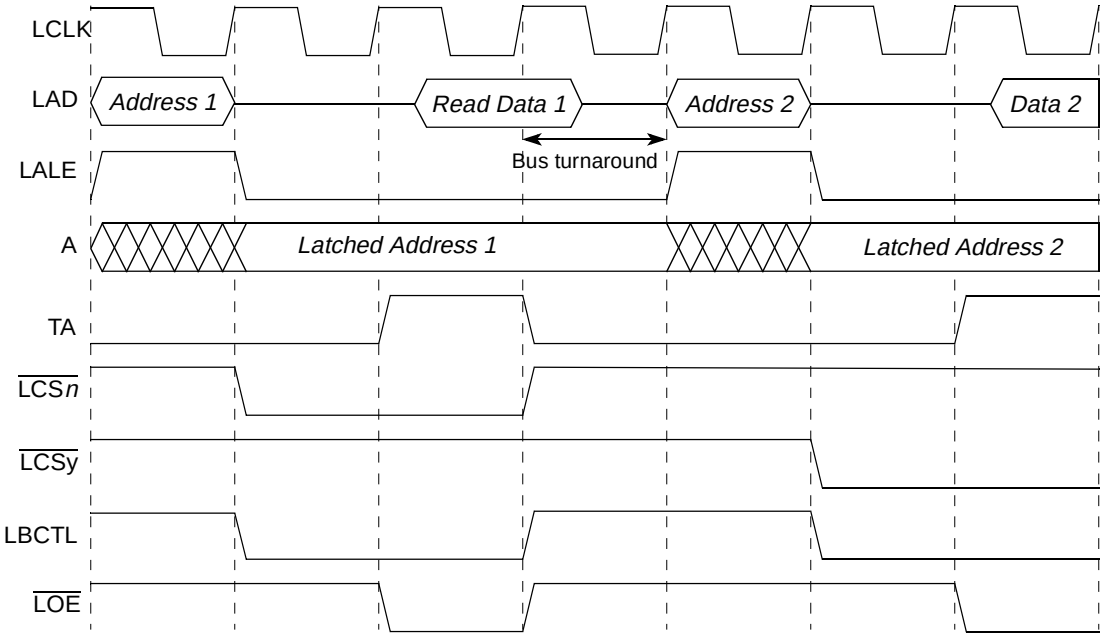


Figure 10-27. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

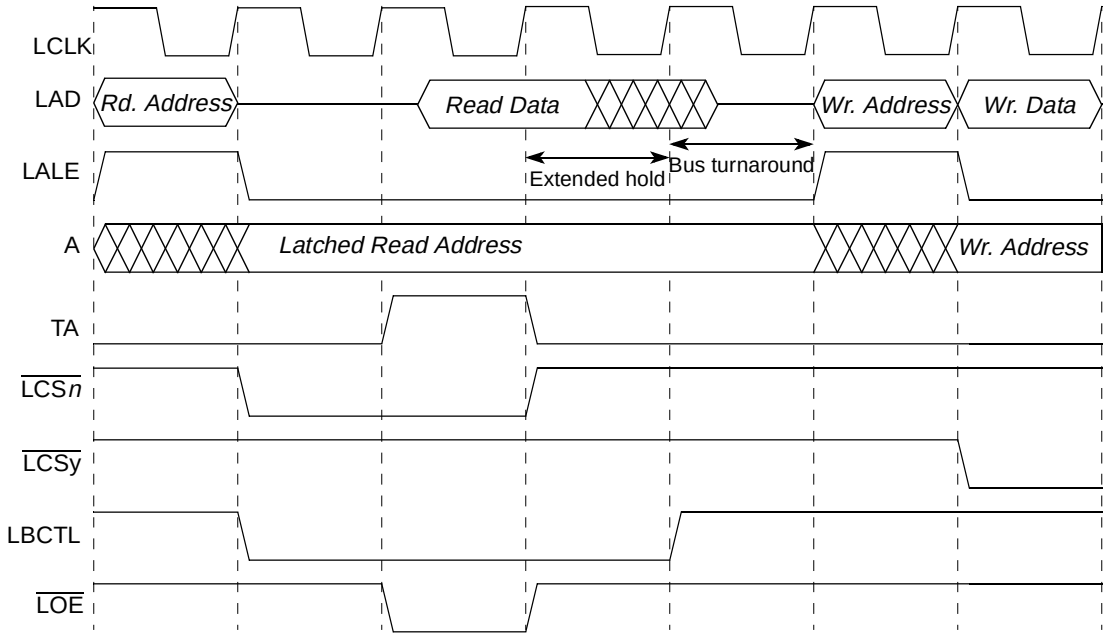


Figure 10-28. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, One-Cycle Extended Hold Time on Reads)

10.4.2.3 External Access Termination ($\overline{\text{LGTA}}$)

External access termination is supported by the GPCM using the asynchronous $\overline{\text{LGTA}}$ input signal, which is synchronized and sampled internally by the local bus. If, during assertion of $\overline{\text{LCS}n}$, 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 case of read cycle, the device still must drive data as long as $\overline{\text{LOE}}$ is asserted.

The user selects whether transfer acknowledge is generated internally or externally ($\overline{\text{LGTA}}$) by programming 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 in Figure 10-29.

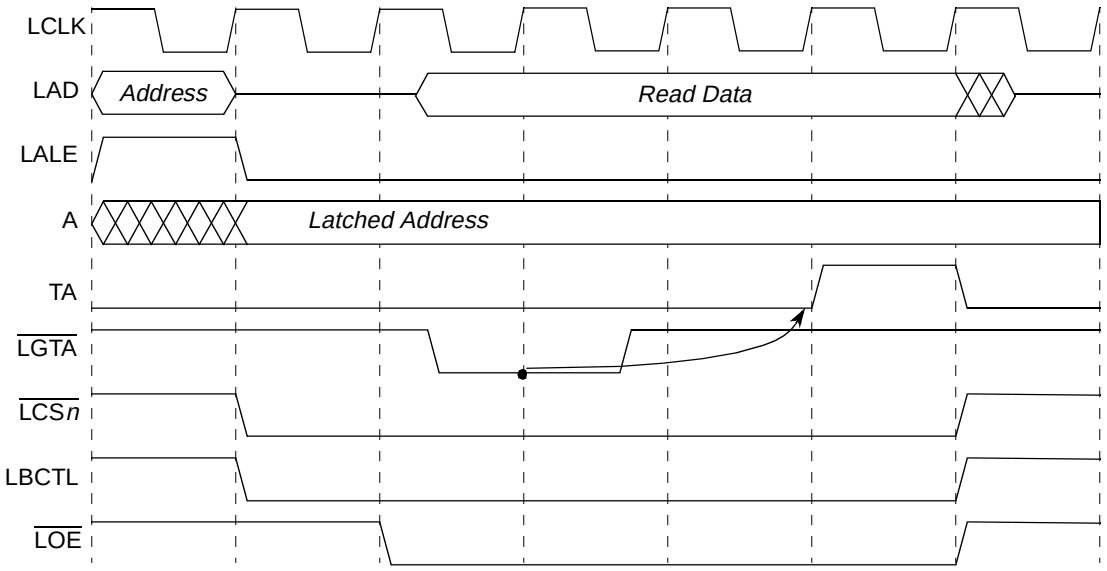


Figure 10-29. External Termination of GPCM Access

10.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{\text{LCS0}}$ is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset, $\overline{\text{LCS0}}$ is asserted for every local bus access until BR0 or OR0 is reconfigured.

The boot chip-select also provides a programmable port size, which is configured during reset. The boot chip-select does not provide write protection. $\overline{\text{LCS0}}$ operates this way until the first write to OR0 and it can be used as any other chip-select register after the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only with a hardware reset. Table 10-24 describes the initial values of the boot bank in the memory controller.

Table 10-24. Boot Bank Field Values After Reset

Register	Field	Setting
BR0	BA PS DECC WP MSEL ATOM V	0000_0000_0000_0000_0 From RCWH[ROMLOC] 00 0 000 00 1

Table 10-24. Boot Bank Field Values After Reset (continued)

Register	Field	Setting
OR0	AM	0000_0000_0000_0000_0
	BCTLD	0
	CSNT	1
	ACS	11
	XACS	1
	SCY	1111
	SETA	0
	TRLX	1
	EHTR	1
	EAD	1

10.4.3 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 ($\overline{\text{LCS}}n$, $\overline{\text{LBS}}$[0:1] and $\overline{\text{LGPL}}$[0:5]) for a given clock cycle. Each word in the RAM array provides bits that allow a memory access to be controlled with a resolution of up to one quarter of the external bus clock period on the byte-select and chip-select lines. A gap of 2 dead LCLK cycles is present on the UPM interface between UPM transactions. Figure 10-30 shows the basic operation of each UPM.

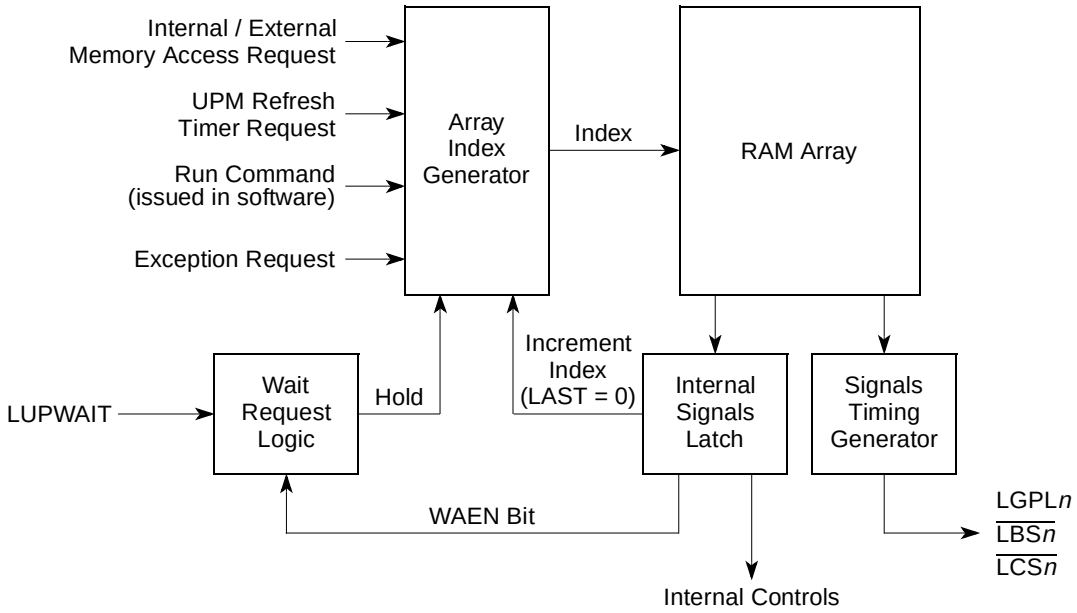


Figure 10-30. User-Programmable Machine Functional Block Diagram

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

10.4.3.1 UPM Requests

A special pattern location in the RAM array is associated with each possible UPM request. An internal device's request for a memory access initiates one of the following patterns (M*n*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 10-31 and Table 10-25 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (M*n*MR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

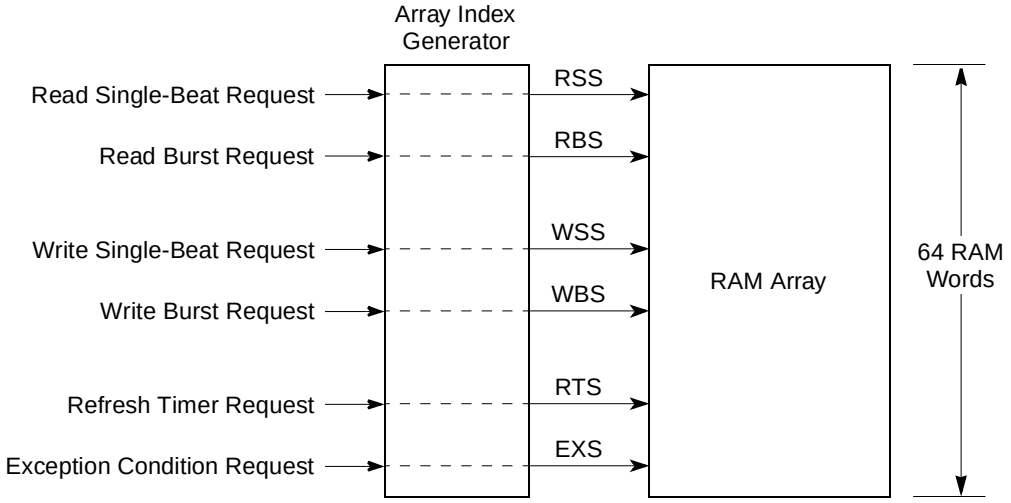


Figure 10-31. RAM Array Indexing

Table 10-25. UPM Routines Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

10.4.3.1.1 Memory Access Requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly four double words regardless of port size. For 16-bit accesses, the burst cycle starts with one transfer start but ends after sixteen transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting OR*n*[BI]. Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

10.4.3.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 10-32 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

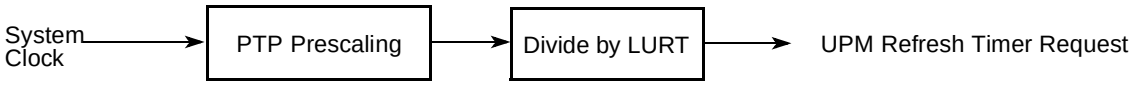


Figure 10-32. Memory Refresh Timer Request Block Diagram

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*n*MR[RFEN] bits are set.

10.4.3.1.3 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.

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*n*MR[OP] = 11 and accessing UPM*n* memory region with any write transaction that hits the corresponding UPM machine. M*n*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.

10.4.3.1.4 Exception Requests

When the LBC under UPM control initiates an access to a memory device and an exception occurs (bus monitor time-out), the UPM provides a mechanism by which memory control signals can meet the device's timing requirements without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

10.4.3.2 Programming the UPMs

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program the UPMs:

1. Set up BR*n* and OR*n* registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT, and MAMR[RFEN] if refresh is required.
4. Program M*n*MR.

Patterns are written to the RAM array by setting M*n*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*n*MR[OP] = 10).

M*n*MR / MDR registers should not be updated while dummy read/write access is still in progress. If the M*n*MR[MAD] is incremented, the previous dummy transaction is already completed. To enforce proper

ordering between updates to the M*n*MR/MDR register and the dummy accesses to the UPM memory region, two rules must be followed:

- Because the result of any update to the M*n*MR/MDR register must be in effect before the dummy read or write to the UPM region, a write to M*n*MR/MDR should be followed immediately by a read of M*n*MR/MDR.
- The UPM memory region should have the same MMU settings as the memory region containing the M*n*MR configuration register; both should be mapped by the MMU as Cache-Inhibited and Guarded. This prevents the e300c2 core from reordering a read of the UPM memory around the read of M*n*MR. When the programming of the UPM array is complete, the MMU setting for the associated address range can be set to the proper mode for normal operation, such as cacheable and copyback.

For proper signalling, the following guidelines must be followed while programming UPM RAM words:

- For UPM reads, program UTA and LAST in the same or consecutive RAM words.
- For UPM burst reads, program last UTA and LAST in the same or consecutive RAM words.
- For UPM writes, program UTA and LAST in the same RAM word.
- For UPM burst writes, program last UTA and LAST in the same RAM word.

10.4.3.2.1 UPM Programming Example (Two Sequential Writes to the RAM Array)

The following example further illustrates the steps required to perform two writes to the RAM array at non-sequential addresses assuming that the relevant BR*x* and OR*x* registers have been previously set up:

1. Program M*n*MR for the first write (with the desired RAM array address).
2. Write pattern/data to MDR to ensure that the M*n*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*n*MR register if step 2 is not performed.)
4. Perform a dummy write transaction. (Write transaction can now be performed.)
5. Read/check M*n*MR[MAD]. If incremented, the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program M*n*MR for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the M*n*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*n*MR[MAD]. If incremented, the previous dummy write transaction is completed.

Note that if step 1 (or 6) and 2 (or 7) are reversed, step 3 (or 8) is replaced by the following:

- Read M*n*MR to ensure that the M*n*MR has already been updated with the desired configuration.

10.4.3.2.2 UPM Programming Example (Two Sequential Reads from the RAM Array)

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (M*n*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*x* and OR*x* registers have been previously set up:

1. Program M*n*MR for the first read with the desired RAM array address.
2. Read M*n*MR to ensure that the M*n*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*n*MR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program M*n*MR for the second read with the desired RAM array address.
7. Read M*n*MR to ensure that the M*n*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*n*MR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

10.4.3.3 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 10-33 and Figure 10-34. If LCRR[CLKDIV] = 2, the bus cycle comprises only two active phases, T1 and T3, which correspond with the first and second halves of the bus clock cycle, respectively. However, if LCRR[CLKDIV] = 4 or 8, four phases, T1–T4, define four quarters of the bus clock cycle. Because T2 and T4 are inactive when LCRR[CLKDIV] = 2, UPM ignores signal timing programmed for assertion in either of these phases in the case LCRR[CLKDIV] = 2.

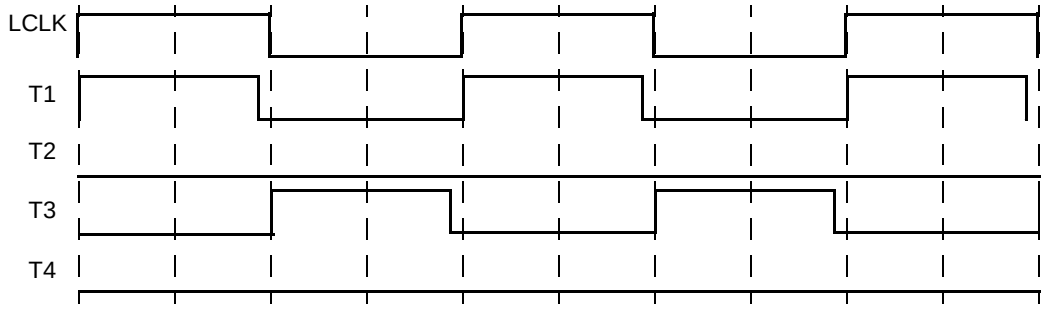


Figure 10-33. UPM Clock Scheme for LCRR[CLKDIV] = 2

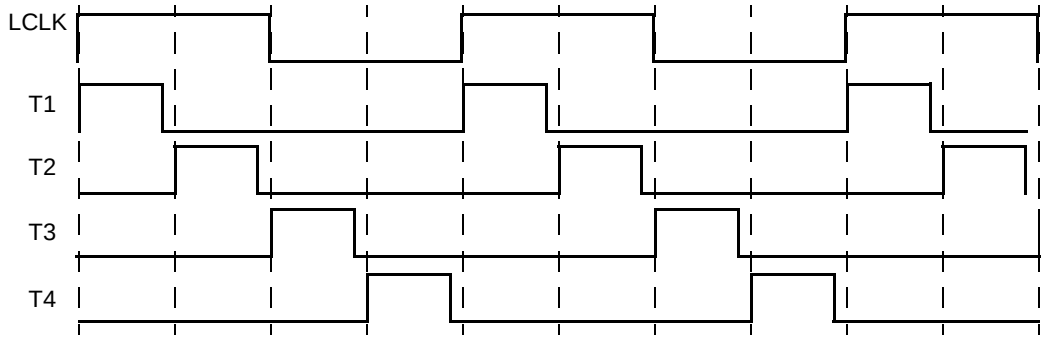


Figure 10-34. UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8

10.4.3.4 UPM RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 10-35. 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.

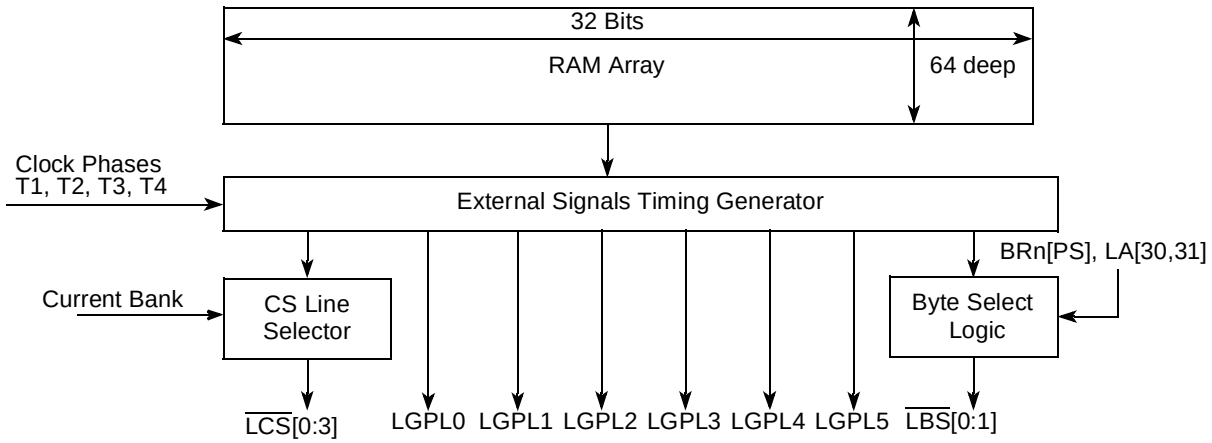


Figure 10-35. UPM RAM Array and Signal Generation

10.4.3.4.1 UPM RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 10-36 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 $\overline{\text{LBS}}$[0:1] 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.

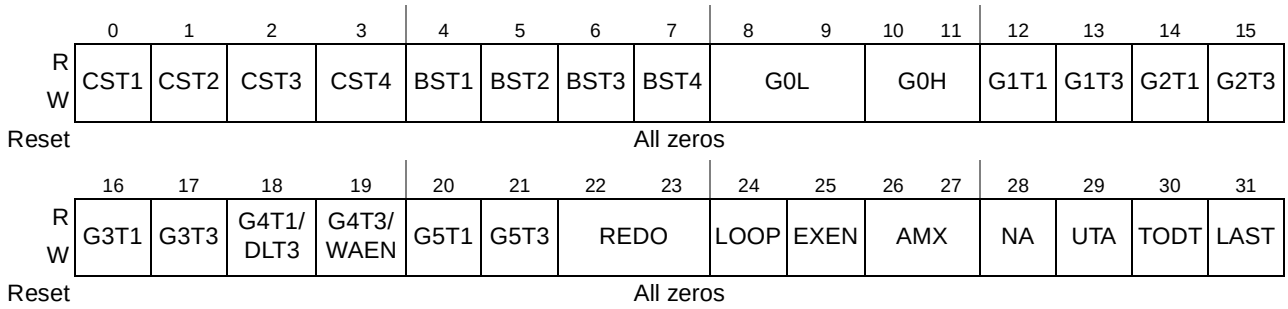


Figure 10-36. UPM RAM Word Field Descriptions

Table 10-26 describes RAM word fields.

Table 10-26. UPM 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 timing1. 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 LA[24:25].
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 LA[24:25]. 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 LA[24:25].
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 LA[24:25]. Ignored when LCRR[CLKDIV] = 2.

Table 10-26. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
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*n*MR[G0CL] 01 Reserved 10 Asserted 11 Negated
10–11	G0H	General purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by M*n*MR[G0CL] 01 Reserved 10 Asserted 11 Negated
12	G1T1	General purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).
13	G1T3	General purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
15	G2T3	General purpose line 2 timing 3. Defines the state (0 or 1) of LGPL2 during bus clock quarter phases 3 and 4 (second half phase).
16	G3T1	General purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).
18	G4T1/DLT3	General purpose line 4 timing 1/delay time 3. The function of this bit is determined by M*n*MR[GPL4]. If M*n*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*n*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*n*MR[GPL4]. If M*n*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*n*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 3 external signals logical values occurs until LUPWAIT is detected as being negated.
20	G5T1	General purpose line 5 timing 1. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 1 and 2 (first half phase).
21	G5T3	General purpose line 5 timing 3. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 3 and 4 (second half phase).

Table 10-26. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
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*n*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. **Note:** AMX must not be changed from its previous value in any RAM word which begins a loop.
25	EXEN	Exception enable. Allows branching to an exception pattern at the exception start address (EXS). When an internal bus monitor time-out exception is recognized and EXEN in the RAM word is set, the UPM branches to the special exception start address (EXS) and begins operating as the pattern defined there specifies. The user should provide an exception pattern to negate signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should negate RAS and CAS to prevent data corruption. If EXEN = 0, exceptions are ignored by UPM (but not by the local bus) and execution continues. After the UPM branches to the exception start address, it continues reading until the LAST bit is set in the RAM word. 0 The UPM continues executing the remaining RAM words, ignoring any internal bus monitor time-out. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected.
26–27	AMX	Address multiplexing. Determines the source of LAD[0:15] during a LALE phase. Any change in the AMX field initiates a new LALE (address) phase. 00 LAD[0:15] (and in conjunction with LA[16:25]) is the non-multiplexed address. For example, column address. 01 Reserved 10 LAD[0:15] (and in conjunction with LA[16:25]) is the address multiplexed according to M*n*MR[AM]. For example, row address. 11 LAD[0:15] (and in conjunction with LA[16:25]) is the contents of MAR. Used, for example, to initialize a mode. **Note:** AMX must not be changed from its previous value in any RAM word which begins a loop. **Note:** Source ID debug mode is supported only 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 LA[21:25] is 1 or 2 for port sizes of 8-bits and16-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.

Table 10-26. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
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 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*n*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*n*MR[DS*n*]. 0 The UPM continues executing RAM words. 1 Indicates the last RAM word in the program. Service to the UPM request is done after this cycle concludes.

10.4.3.4.2 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 10-37 shows how UPMs control $\overline{\text{LCS}n}$ signals.

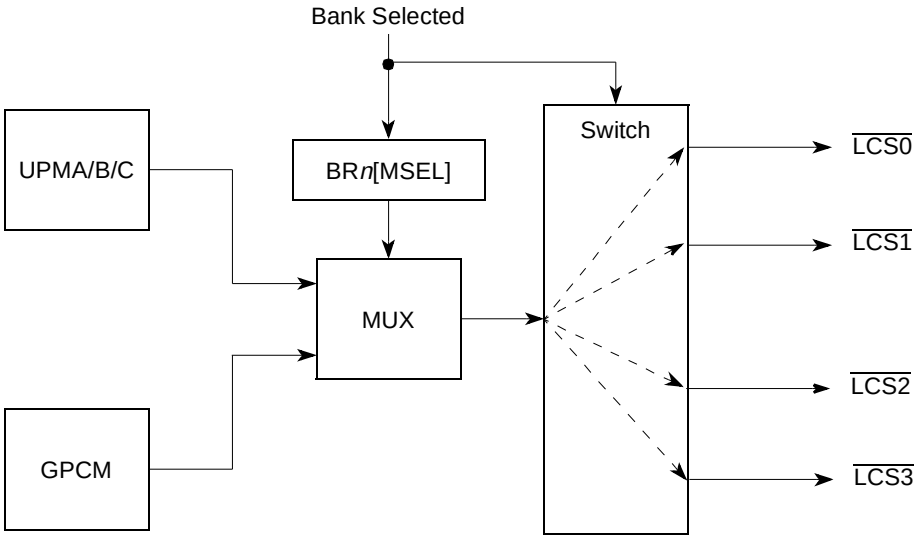


Figure 10-37. $\overline{\text{LCS}n}$ Signal Selection

10.4.3.4.3 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 $\overline{\text{LBS}}$[0:1] signal. The timing of both byte-select

signals is specified in the RAM word. However, $\overline{\text{LBS}}$[0:1] are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

Figure 10-38 shows how UPMs control $\overline{\text{LBS}}$[0:1].

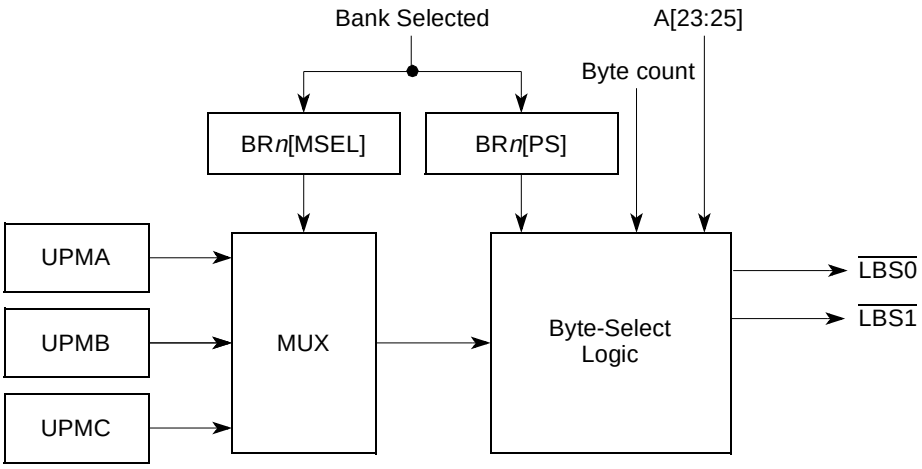


Figure 10-38. $\overline{\text{LBS}}$ Signal Selection

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. For a UPM refresh timer request, both $\overline{\text{LBS}}$[0:1] signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, the $\overline{\text{LBS}}$[0:1] signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

10.4.3.4.4 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*n*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.

10.4.3.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 10-27. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken: LAST and LOOP must not be set together.

Table 10-27. M*n*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

10.4.3.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

10.4.3.4.7 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*n*MR[AM] field, or driving the MAR contents on the address signals. In all cases, LA[21:25] 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 10-28 shows how M*n*MR[AM] settings affect address multiplexing when the RAM word AMX = 10. The 10 msbs of the LAD[0:15] bus during an address phase are driven with zero in the AMX = 10 case.

Table 10-28. UPM Address Multiplexing

AM	LAD[0:15], LA[16:25]} as Address Signals	A0–A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25
000	Signal driven on external signal when address multiplexing is enabled—RAM word AMX = 10	0	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		0	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		0	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		0	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		0	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		0	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Note that any change to AMX from one RAM word to the next RAM word executed results in an address phase on {LAD[0:15], LA[16:25]} with the assertion of LALE for the number of cycles set for LALE in the OR*n* and LCRR registers. LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

NOTE

AMX must not be changed from its previous value in any RAM word which begins a loop.

10.4.3.4.8 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*n*MR[GPL*n*4DIS], determines when the data input is sampled by the LBC as follows:

- If M*n*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 10-39 shows how data sampling is controlled by the UPM.

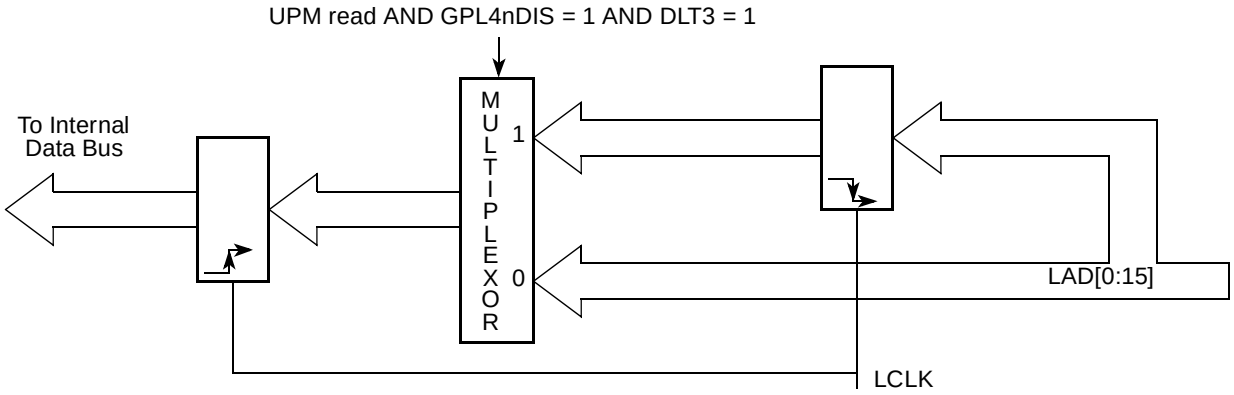


Figure 10-39. UPM Read Access Data Sampling

10.4.3.4.9 $\overline{\text{LGPL}}$[0:5] Signal Negation (LAST)

When the LAST bit is read in a RAM word, the current UPM pattern is terminated at the end of the current cycle. On the next cycle (following LAST) all UPM signals are negated unconditionally (driven to logic one), unless there is a back-to-back UPM request pending. In this case, the signal values for the cycle following the one in which the LAST bit was set are taken from the first RAM word of the pending UPM routine.

10.4.3.4.10 Wait Mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if LAST = 1 in the same RAM word.

Synchronization of LUPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If LUPWAIT is asserted and WAEN = 1 in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the previous RAM word. Setting the WAEN bit for the first RAM word does not have any effect because the first RAM word signifies the start of a new bus cycle and the initial values of the signals driven onto the bus should be present. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 10-40 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 $\overline{\text{LGPL1}}$ 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.

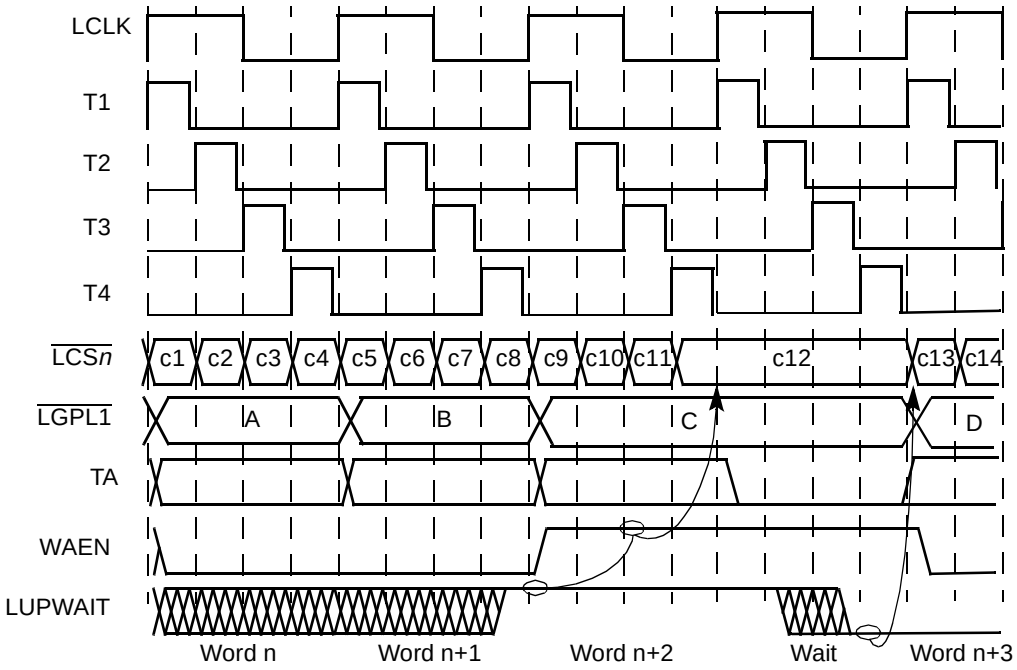


Figure 10-40. Effect of LUPWAIT Signal

10.4.3.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both WAEN = 1 and UTA = 1 simultaneously.

However, programming WAEN = 1 and UTA = 1 in the same RAM word allows the UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether the UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to effect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

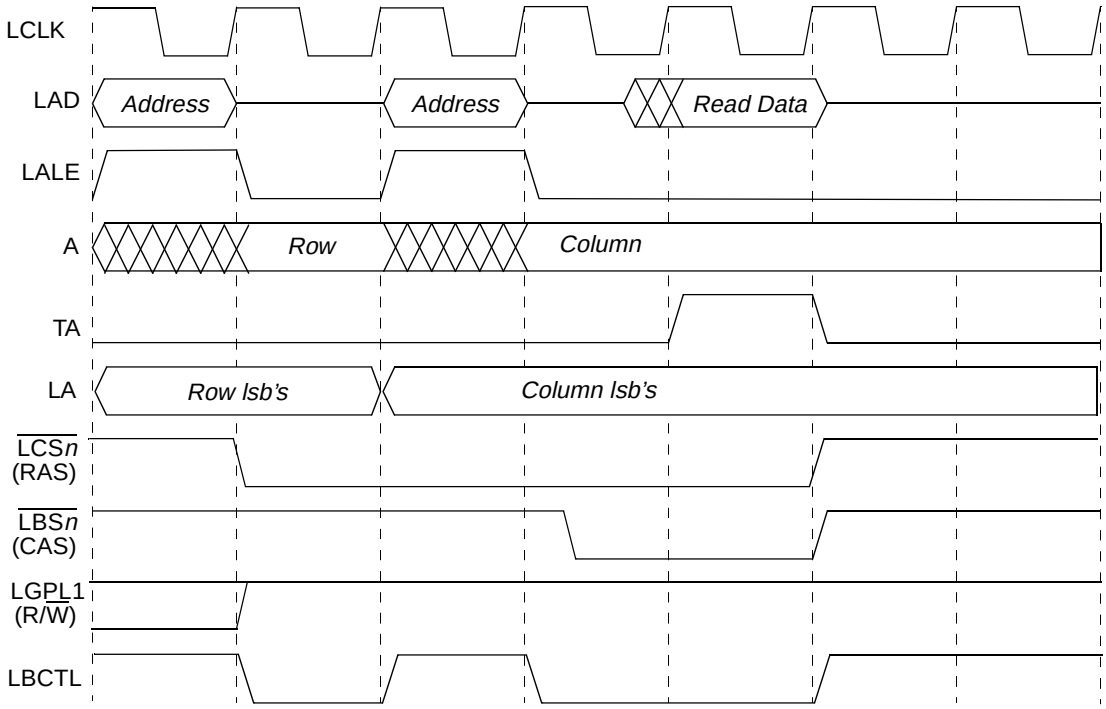
10.4.3.6 Extended Hold Time on Read Accesses

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 access 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 turnaround cycle.

10.4.3.7 Memory System Interface Example Using UPM

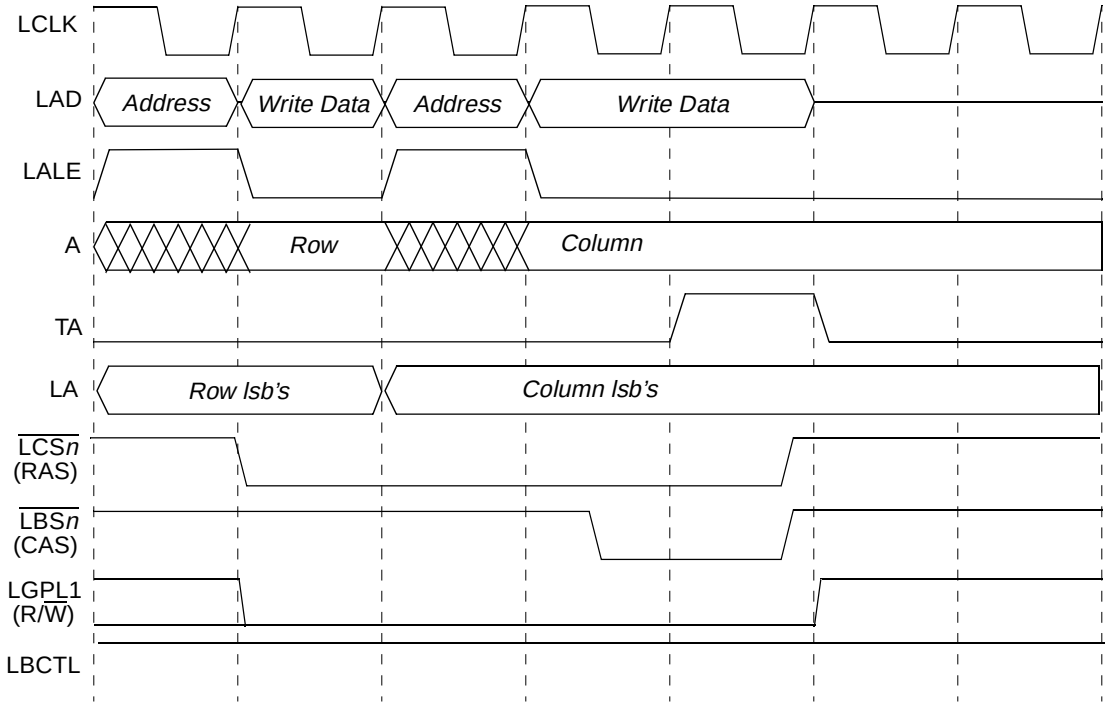
Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section presents timing diagrams for various UPM configurations, using fast-page mode DRAM as

an example, with LCRR[CLKDIV] = 4 or 8. These examples may not represent the timing necessary for any specific device used with the LBC. Here, LGPL1 is programmed to drive R/$\overline{W}$ of the DRAM, although any LGPL*n* signal can be used for this purpose.



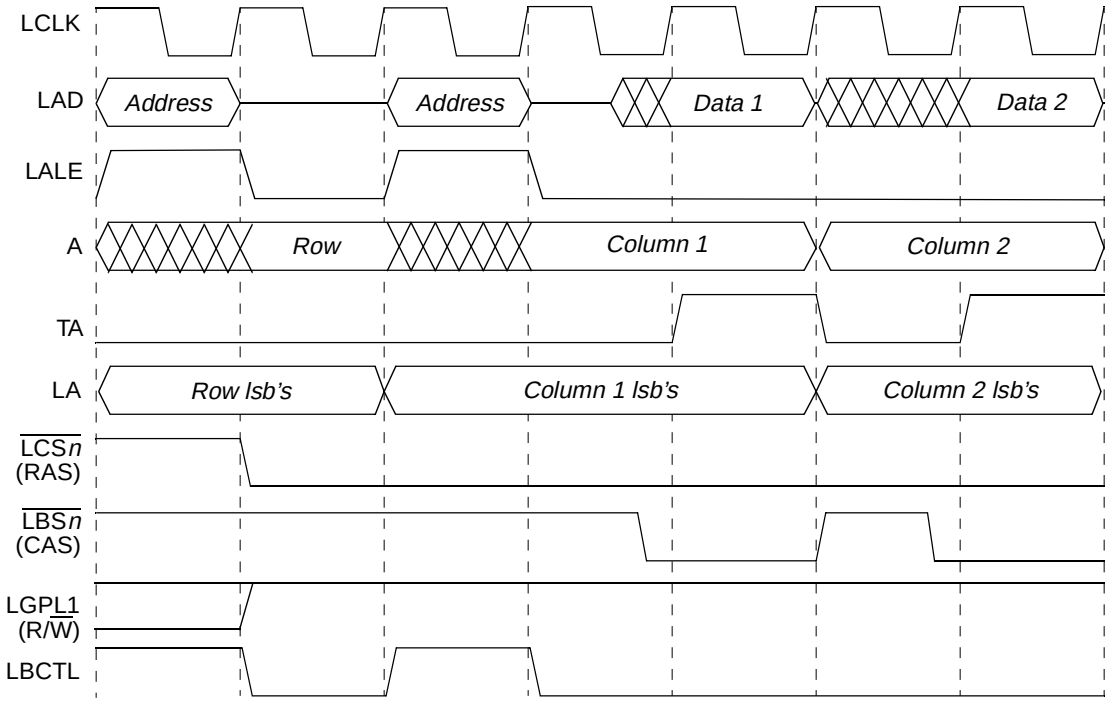
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 0 0 0	0 0 0 0	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit 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
	RSS	**RSS+1**	**RSS+1**	**RSS+2**	

Figure 10-41. Single-Beat Read Access to FPM DRAM



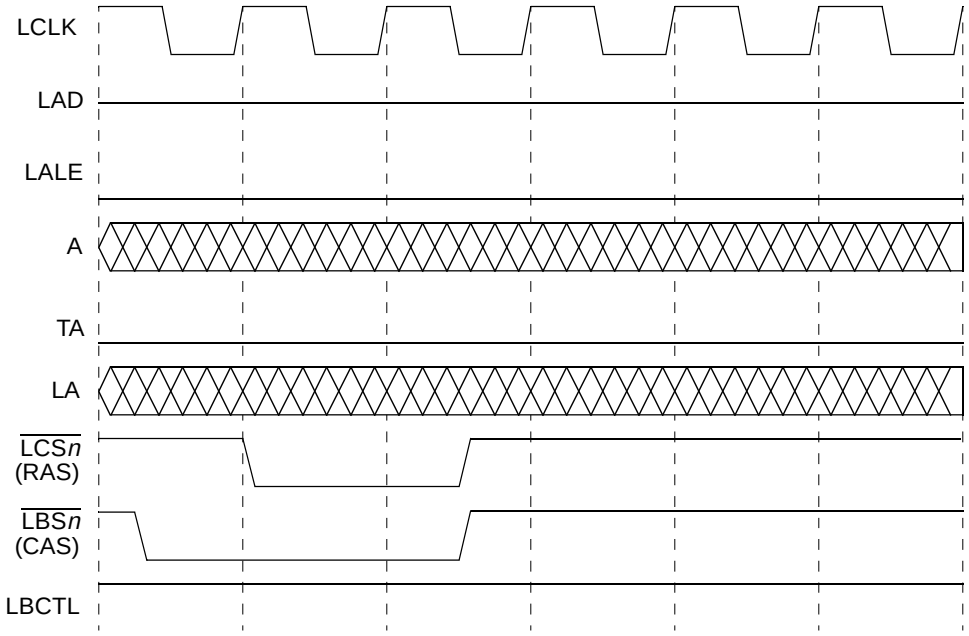
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 0 0	0 0 0 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	0 0		0 0	0 0	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit 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 10-42. Single-Beat Write Access to FPM DRAM



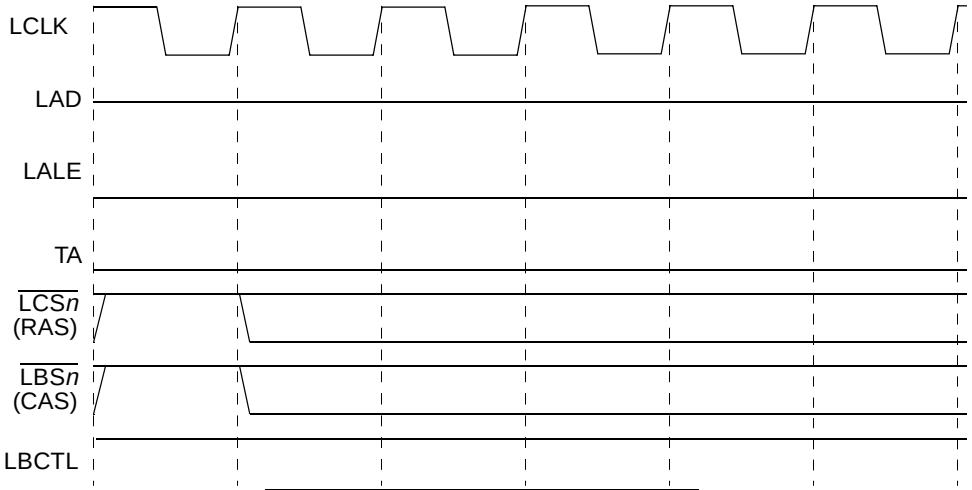
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 1 0	0 0 0 0	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1						Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	1 1	Bit 12 Bit 13
g2t1 g2t3						Bit 14 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		1 0	1 1	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 1 1 0 0	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	RBS		**RBS+1**	**RBS+2**	**RBS+3**	

Figure 10-43. Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)



cst1 cst2 cst3 cst4	1 1 1 1	0 0 0 0	0 0 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 0 0 0	0 0 0 0	0 0 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1				Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3				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	0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	PTS	**PTS+1**	**PTS+2**	

Figure 10-44. Refresh Cycle (CBR) to FPM DRAM



cst1 cst2 cst3 cst4	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1		Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3		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	Bit 24 Bit 25
amx0 amx1 na uta todt last	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	EXS	

Figure 10-45. Exception Cycle

10.5 Initialization/Application Information

These sections provide detailed information on interfacing the LBC to peripheral devices.

10.5.1 Interfacing to Peripherals in Multiplexed Address/Data Mode

The local bus can be used either with separate address- and data buses or with a multiplexed address/data bus. These sections provide guidelines for interfacing peripherals in the multiplexed mode.

10.5.1.1 Multiplexed Address/Data Bus and Unmultiplexed Address Signals

To save signals on the local bus, address and data are multiplexed onto the same 16-bit bus. An external latch is needed to demultiplex the 16-bit MSB address and, together with LA[16:25], to reconstruct the original address. No external intelligence is needed, because LALE provides the correct timing to control a standard logic latch. The LAD signals can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus start with an address phase, where the LBC drives the transaction address on the LAD signals and asserts the LALE signal. This can be used to latch the address and then the LBC can continue with the data phase.

The LBC supports port sizes of 8 and 16 bits. For devices smaller than 16 bits, transactions must be broken down. For this reason, LA[24:25] are driven unmultiplexed. For 8-bit devices, LA[24:25] should be used and for 16 bit devices, LA24 should be used.

In addition, the LBC supports burst transfers (not in the GPCM machine). LA[21:23] are the burst addresses within a natural 32-byte burst.

All other addresses, A[0:15], must be reconstructed through the latch.

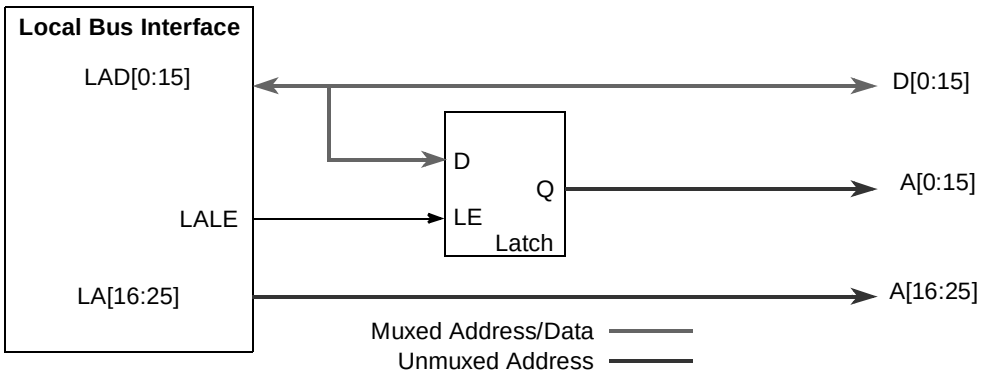


Figure 10-46. Multiplexed Address/Data Bus

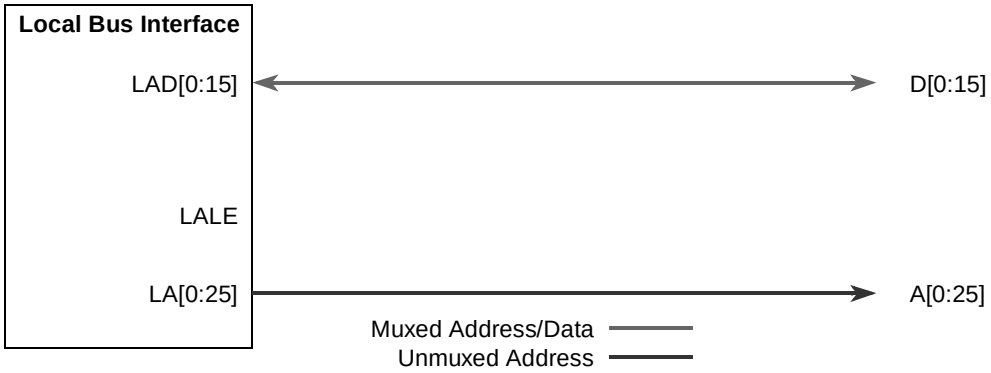


Figure 10-47. Non-Multiplexed Address/Data Bus

10.5.1.2 GPCM Timings

If a system contains a memory hierarchy with high-speed synchronous memories (synchronous SRAM) and lower speed asynchronous memories (FLASH EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations. The GPCM address timings is shown in Figure 10-48.

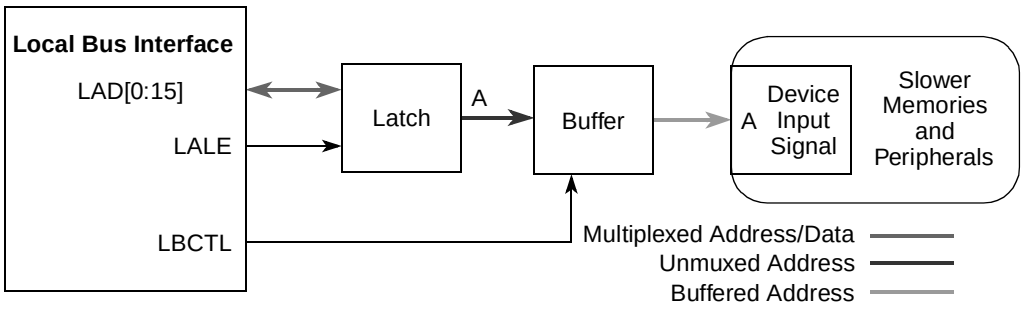


Figure 10-48. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the two propagation delays are in the order of 3–6 ns, so for a 166-MHz bus frequency, $\overline{\text{LCS}}$ should arrive roughly 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time must be considered.

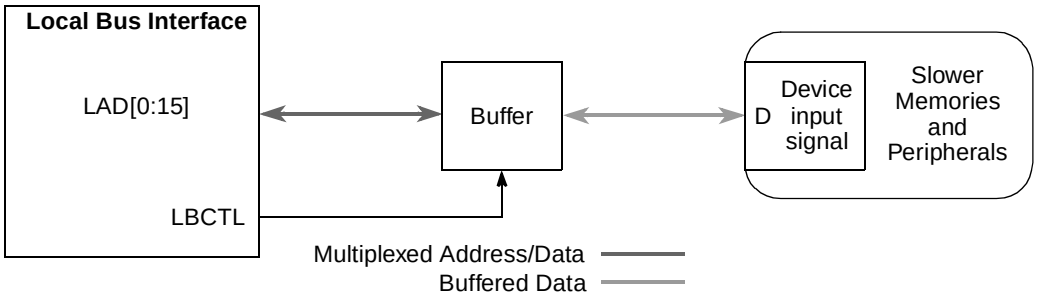


Figure 10-49. GPCM Data Timings

10.5.2 Bus Turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround. The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- UPM cycles with additional address phases

The bus does not change direction for the following cases, so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

10.5.2.1 Address Phase after Previous Read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD. After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices, the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the LBC memory controller ends the bus cycle.

In this case, after the previous cycle ends, LBCTL goes high and changes the direction of the bus transceiver. The LBC then inserts a bus turnaround cycle to avoid contention. The external device has now already placed its data signals in high impedance and no bus contention will occur.

10.5.2.2 Read Data Phase after Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving the bus transceivers in the same direction as during a write. After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The LBC places the LAD signals in high impedance after its t_{dis}(LB). The LBCTL will have its new state after t_{en}(LB) and, because this is an asynchronous input, the transceiver starts to drive those signals after its t_{en}(transceiver) time. The system designer has to ensure, that [t_{en}(LB) + t_{en}(transceiver)] is larger than t_{dis} (LB) to avoid bus contention.

10.5.2.3 UPM Cycles with Additional Address Phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore turning around the bus during one pattern. The LBC automatically inserts a single bus turnaround cycle if the bus (LAD) was previously high impedance for any reason, such as a read, before LALE is driven and LAD is driven with the new address. The turnaround cycle is not inserted on a write, because the bus was already driven to begin with.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

10.5.3 Interface to Different Port-Size Devices

The LBC supports 8- and 16-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 16-bit port must reside on D[0:15], and an 8-bit port must reside on D[0:7]. The local bus always tries to transfer the maximum amount of data on all bus cycles. Figure 10-50 shows the device connections on the data bus.

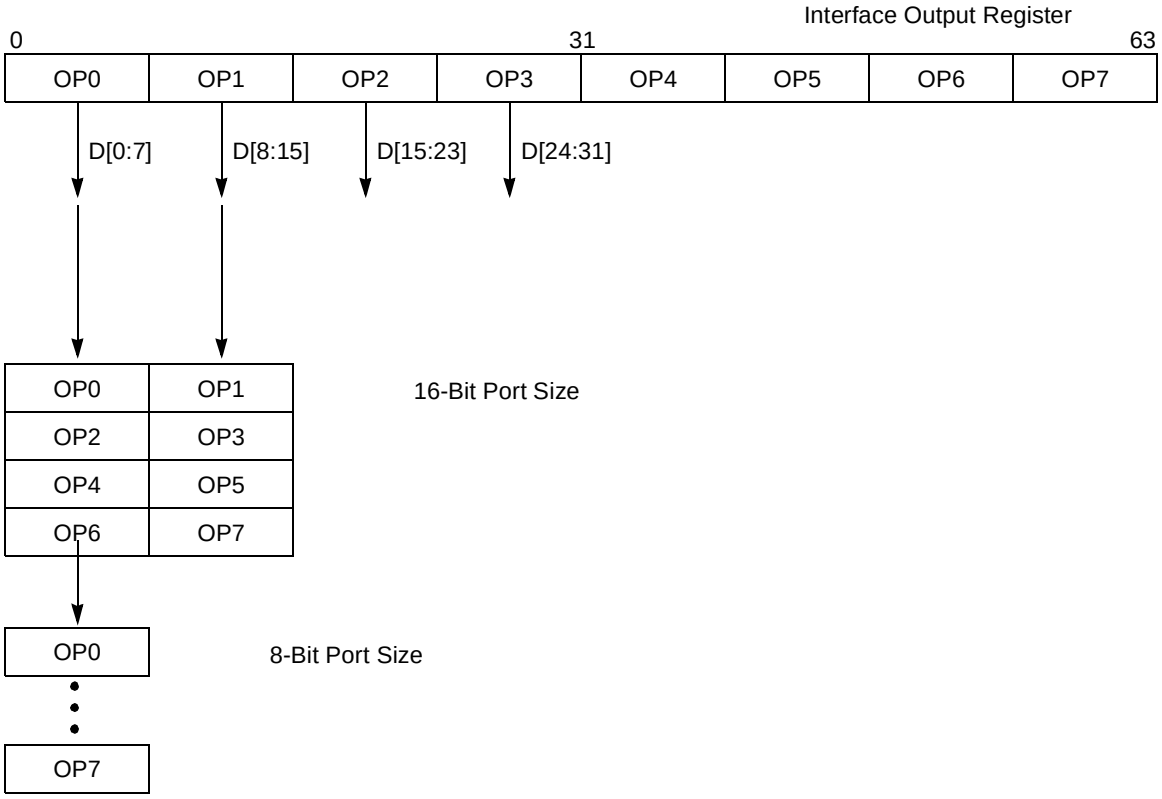


Figure 10-50. Interface to Different Port-Size Devices

Table 10-29 lists the bytes required on the data bus for read cycles.

Table 10-29. Data Bus Requirements For Read Cycle

Transfer Size	Address State [1] A[23–25]	Port Size/Data Bus Assignments		
		16-Bit		8-Bit
		0–7	8–15	0–7
Byte	000	OP0[2]	—[3]	OP0
	001	—	OP1	OP1
	010	OP2	—	OP2
	011	—	OP3	OP3
	100	OP4	—	OP4
	101	—	OP5	OP5
	110	OP6	—	OP6
	111	—	OP7	OP7
Half Word	000	OP0	OP1	OP0
	001	—	OP1	OP1
	010	OP2	OP3	OP2
	100	OP4	OP5	OP4
	101	—	OP5	OP5
	110	OP6	OP7	OP6
Word	000	OP0	OP1	OP0
	100	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 read cycle.

10.5.4 Interfacing to ZBT SRAM

ZBT SRAMs have been designed to optimize the performance in networking applications. This section describes how to interface to ZBT SRAMs. Figure 10-51 shows the connections. The UPM is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. Because ZBT SRAMs are used mostly by performance-critical applications, a 16-bit maximum local bus width is assumed.

ZBT SRAMs allow different configurations. For the local bus, the burst order should be set to linear burst order by tying the mode signal to GND; $\overline{\text{CKE}}$ should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the LBC generates sixteen-beat transactions (for 16-bit ports) the UPM breaks down each burst into four consecutive four-beat bursts. The internal address generator of the LBC generates the new {A21, A22} for the second, third and fourth burst.

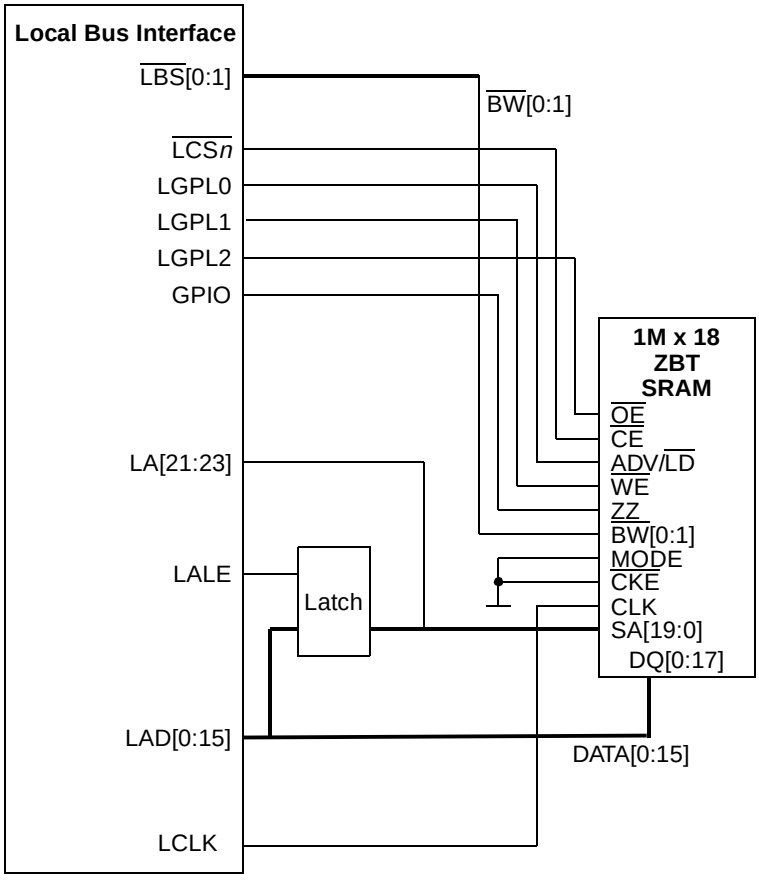


Figure 10-51. Interface to ZBT SRAM

Because linear burst on the SRAM is used, the device itself bursts with the burst addresses of [0:1:2:3]. The local bus always generates linear bursts and expects [0:1:2:3:4:5:6:7:...:15]. Therefore, four consecutive linear bursts of the ZBT SRAM with {A21,A22}={0,0} for the first burst, {A21,A22}={0,1} for the second burst, {A21,A22}={1,0} for the third burst and {A21,A22}={1,1} for the fourth burst, give the desired burst pattern.

The UPM also supports single beat accesses. Because the ZBT SRAM does not support this and always responds with a burst, the UPM pattern must take care that data for the critical beat is provided (for write) or sampled (for read), and that the rest of the burst is ignored (by negating $\overline{WE}$). The UPM controller must wait for the end of the SRAM burst to avoid bus contention with further bus activities.

Chapter 11
Sequencer

11.1 Overview

The I/O sequencer switches transactions among its ports, using a buffer pool to minimize blocking. Figure 11-1 is a block diagram of the I/O sequencer (IOS).

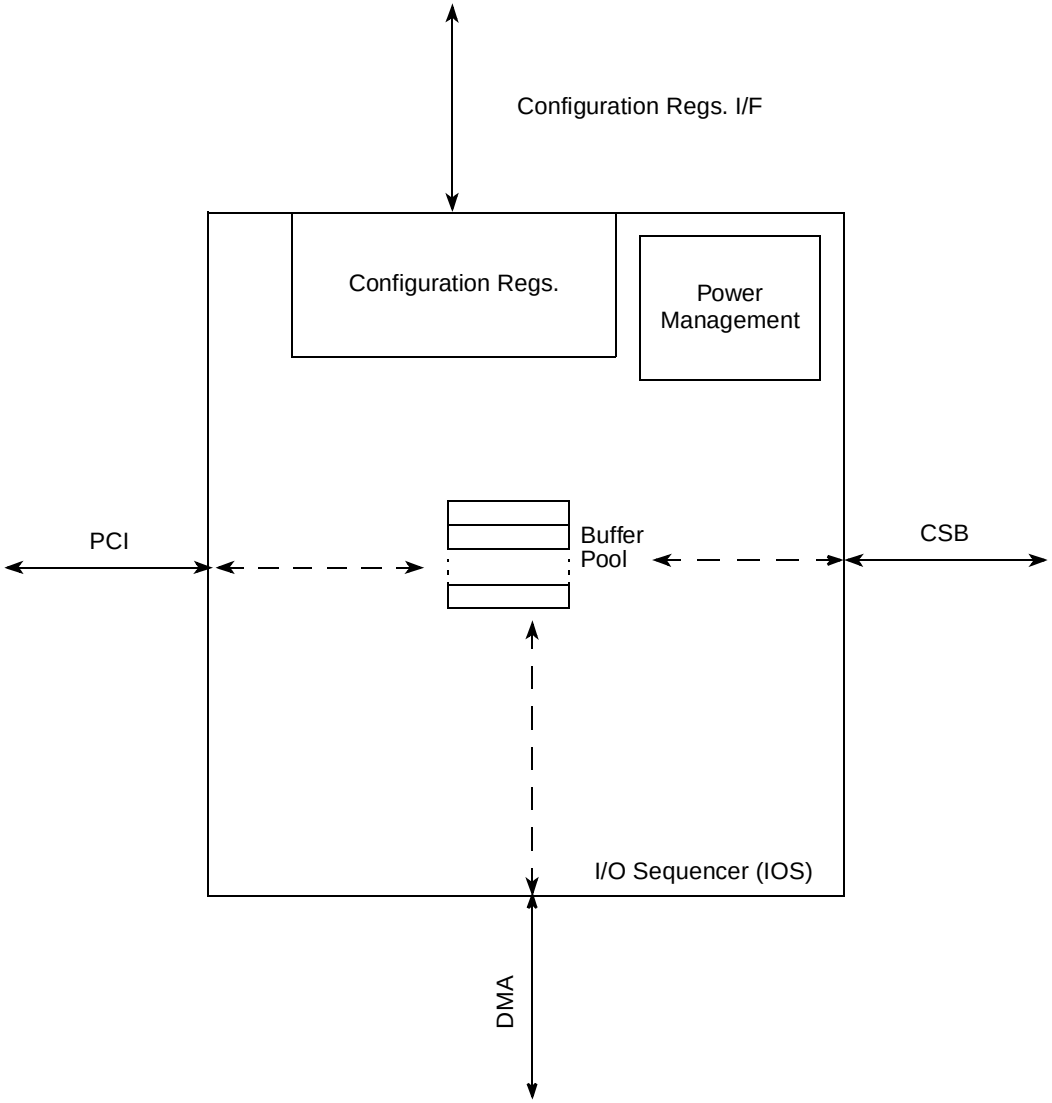


Figure 11-1. I/O Sequencer Block Diagram

11.1.1 Features

The I/O sequencer includes the following features:

- Switches transactions among its ports
- Contains 8 cache-line (32-byte) buffers to allow streaming of PCI transactions
- Performs address translation on outbound PCI transactions

Note that the number of CSB masters allowed to access to PCI outbound windows is restricted to no more than four non-CPU masters or no more than two non-CPU plus the CPU. If this number is exceeded, deadlock can occur on CSB arbitration.

11.2 External Signal Description

The I/O sequencer has no external signals.

11.3 Memory Map/Register Definition

Table 11-1 shows the I/O sequencer memory map.

Table 11-1. Sequencer Memory Map

Offset	Register	Access	Reset	Section/Page
0x00	POTAR0—PCI outbound translation address register 0	R/W	0x0000_0000	11.4.1/11-3
0x08	POBAR0—PCI outbound base address register 0	R/W	0x0000_0000	11.4.2/11-3
0x10	POCMR0—PCI outbound comparison mask register 0	R/W	0x0000_0000	11.4.3/11-4
0x18	POTAR1—PCI outbound translation address register 1	R/W	0x0000_0000	11.4.1/11-3
0x20	POBAR1—PCI outbound base address register 1	R/W	0x0000_0000	11.4.2/11-3
0x28	POCMR1—PCI outbound comparison mask register 1	R/W	0x0000_0000	11.4.3/11-4
0x30	POTAR2—PCI outbound translation address register 2	R/W	0x0000_0000	11.4.1/11-3
0x38	POBAR2—PCI outbound base address register 2	R/W	0x0000_0000	11.4.2/11-3
0x40	POCMR2—PCI outbound comparison mask register 2	R/W	0x0000_0000	11.4.3/11-4
0x48	POTAR3—PCI outbound translation address register 3	R/W	0x0000_0000	11.4.1/11-3
0x50	POBAR3—PCI outbound base address register 3	R/W	0x0000_0000	11.4.2/11-3
0x58	POCMR3—PCI outbound comparison mask register 3	R/W	0x0000_0000	11.4.3/11-4
0x60	POTAR4—PCI outbound translation address register 4	R/W	0x0000_0000	11.4.1/11-3
0x68	POBAR4—PCI outbound base address register 4	R/W	0x0000_0000	11.4.2/11-3
0x70	POCMR4—PCI outbound comparison mask register 4	R/W	0x0000_0000	11.4.3/11-4
0x78	POTAR5—PCI outbound translation address register 5	R/W	0x0000_0000	11.4.1/11-3
0x80	POBAR5—PCI outbound base address register 5	R/W	0x0000_0000	11.4.2/11-3
0x88	POCMR5—PCI outbound comparison mask register 5	R/W	0x0000_0000	11.4.3/11-4

Table 11-1. Sequencer Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xF0	PMCR—Power management control register	R/W	0x0000_0000	11.4.4/11-5
0xF8	DTCR—Discard timer control register	R/W	0x0000_0000	11.4.5/11-6

11.4 Register Descriptions

11.4.1 PCI Outbound Translation Address Registers (POTAR*n*)

The PCI outbound translation address register defines the location of the outbound translation window in the PCI (translated) address space. Figure 11-2 shows the POTAR*n* register fields.

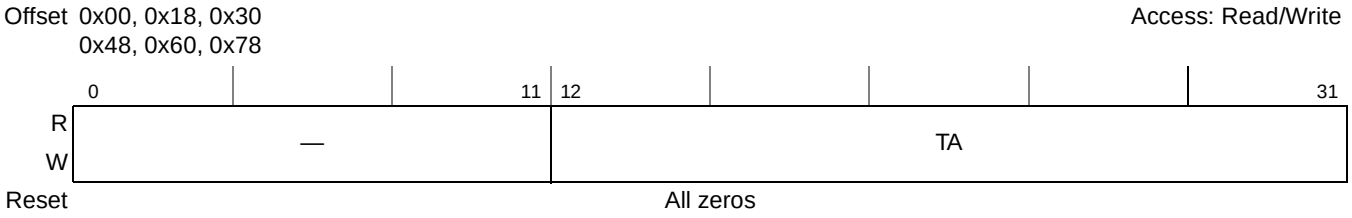


Figure 11-2. PCI Outbound Translation Address Registers (POTAR*n*)

Table 11-2 describes POTAR*n* fields.

Table 11-2. POTAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TA	Translation address. Contains the starting address of the outbound translated address. It also corresponds to the most-significant 20 bits of a 32-bit address. The translation address must be aligned based on the window's size.

11.4.2 PCI Outbound Base Address Registers (POBAR*n*)

The PCI outbound base address register (POBAR*n*) defines the location of the outbound translation window in the local (source) memory space. Figure 11-3 shows the POBAR*n* register fields.

Figure 11-3. PCI Outbound Base Address Registers (POBAR*n*)

Table 11-3 describes POBAR*n* fields.

Table 11-3. POBAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	BA	Base address. This field contains the starting address of the outbound translated window. This field corresponds to the most-significant 20 bits of a 32-bit address.

11.4.3 PCI Outbound Comparison Mask Registers (POCMR*n*)

The PCI outbound comparison mask register (POCMR*n*) defines the size and destination of the outbound translation window. It also defines some properties of the window in the PCI address space. See Section 11.5.1, “Transaction Forwarding,” for more information. Figure 11-4 shows the POCMR*n* register fields.

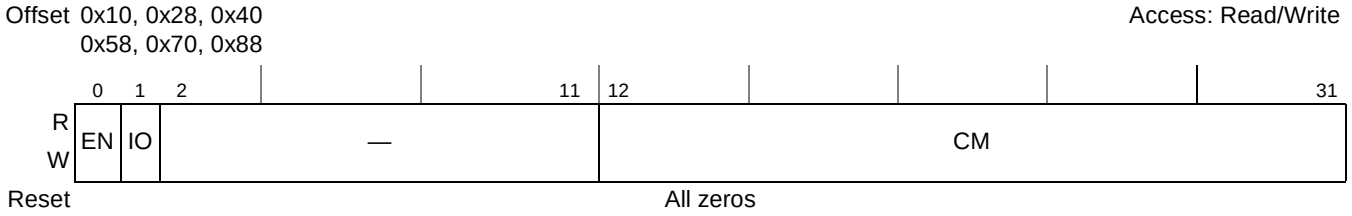


Figure 11-4. PCI Outbound Comparison Mask Registers (POCMR*n*)

Table 11-4 describes the bit settings of the POCMR*n* register.

Table 11-4. POCMR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables the address translation window. 0 Address translation is disabled for this window. 1 Address translation is enabled for this window. Local addresses that match the definition of the window will be recognized by the device and translated to the PCI memory space.
1	IO	I/O space. Determines whether the window is mapped to the PCI memory space or PCI I/O space. 0 Memory space 1 I/O space
2–11	—	Reserved, should be cleared.

Table 11-4. POCMR*n* Field Descriptions (continued)

Bits	Name	Description
12–31	CM	Comparison mask. Contains the size of the translation window. The bits that are 1 in this field indicate bits of the transaction address that should be matched to the value in the PCI outbound base address register and translated in case of a match. This field corresponds to the most-significant 20 bits of a 32-bit address.

Value	Meaning	Value	Meaning
0000_0000_0000_0000_0000	4 GB	1111_1111_1110_0000_0000	2 MB
1000_0000_0000_0000_0000	2 GB	1111_1111_1111_0000_0000	1 MB
1100_0000_0000_0000_0000	1 GB	1111_1111_1111_1000_0000	512 KB
1110_0000_0000_0000_0000	512 MB	1111_1111_1111_1100_0000	256 KB
1111_0000_0000_0000_0000	256 MB	1111_1111_1111_1110_0000	128 KB
1111_1000_0000_0000_0000	128 MB	1111_1111_1111_1111_0000	64 KB
1111_1100_0000_0000_0000	64 MB	1111_1111_1111_1111_1000	32 KB
1111_1110_0000_0000_0000	32 MB	1111_1111_1111_1111_1100	16 KB
1111_1111_0000_0000_0000	16 MB	1111_1111_1111_1111_1110	8 KB
1111_1111_1000_0000_0000	8 MB	1111_1111_1111_1111_1111	4 KB
1111_1111_1100_0000_0000	4 MB		

All others values are Reserved

11.4.4 Power Management Control Register (PMCR)

PMCR provides control of system-level low-power modes. Figure 11-5 shows the PMCR register fields.

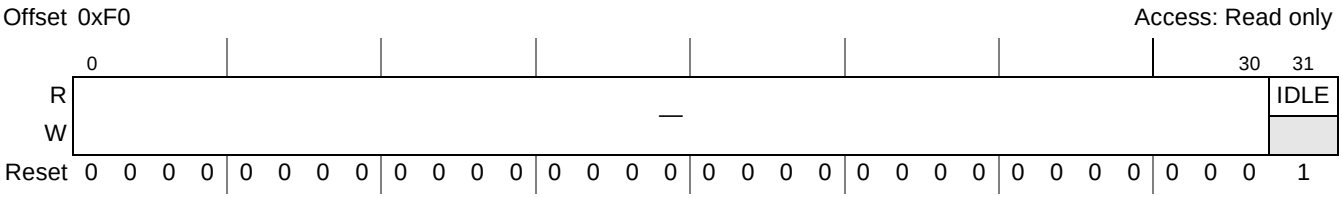


Figure 11-5. Power Management Control Register (PMCR)

Table 11-5 describes PMCR fields.

Table 11-5. PMCR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	IDLE	This read-only bit indicates that the logic controlled by the power management function is idle. This bit could be used as an indication that it is okay to disable the clocks to this complex. 0 Logic is active. 1 Logic is idle. There are no outstanding transactions in the IOS, the DMA, or the PCI port.

11.4.5 Discard Timer Control Register (DTCR)

DTCR configures the discard timer, which is used to place a time limit on PCI delayed read transactions from non-prefetchable memory. Although prefetched reads may be discarded whenever the IOS is full and needs to allocate another buffer, other delayed reads must not be discarded until the originator actually receives the data. The DTCR is used to release stuck buffers in case of malfunctioning or disconnected masters that never come back to read the data they requested

Figure 11-6 shows the DTCR register fields.

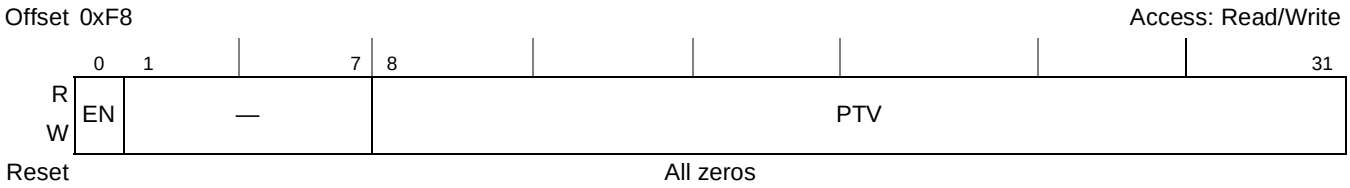


Figure 11-6. Discard Timer Control Register (DTCR)

Table 11-6 describes DTCR fields.

Table 11-6. DTCR Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables the discard timer. 0 Disabled 1 Enabled
1–7	—	Reserved
8–31	PTV	Preload timer value (PTV). This field contains the preload value for the discard timer. PCI delayed reads from non-prefetchable address space are discarded after (2^{24} – PTV) internal clock cycles if the master has not repeated the transaction. 0xFFFFFF is not valid for PTV. For example, to discard a delayed completion if the PCI master has not repeated the transaction in 2^{15} PCI clocks, • Assuming the internal frequency is twice the PCI frequency • The PTV should equal $2^{24} – 2^{16}$ (0xFF0000).

11.5 Functional Description

The IOS is a three-port switch with buffering. Each port has master and slave interfaces. When a port masters a transaction, the transaction attributes are stored in a buffer and the IOS generates a transaction to the slave interface of the destination port. The data is also buffered between the ports. The IOS contains 8 cache line (32-byte) transaction buffers, some of which are reserved for specific types of transactions.

The address and data phases of the transactions are independent. The data phases of the transactions are not required to be in order.

11.5.1 Transaction Forwarding

Although the ports use a similar interface, the I/O sequencer is not actually symmetrical. The transaction forwarding from each source is explained in the following sections.

11.5.1.1 Transactions from the Coherency System Bus (CSB) Port

Transactions from the CSB port are forwarded as follows:

- If the address matches the 12-byte PCI controller software configuration memory space of the PCI controller, the transaction is forwarded to the PCI port. These address values are configuration options of the I/O sequencer. See Table 13-3 for more information on PCI controller software configuration memory space.
- If the address matches the DMA register memory space, the transaction is forwarded to the DMA port.
- If the address hits any of the outbound translation windows, the transaction is forwarded to the PCI port, with the address translated. See Section 11.5.2, “PCI Outbound Address Translation,” for more information.

11.5.1.2 Transactions from the PCI Port

Transactions from the PCI port are forwarded as follows:

- If the address matches the DMA register memory space, the transaction is forwarded to the DMA port.
- All other transactions are forwarded to the CSB port.

11.5.1.3 Transactions from the DMA Port

Transactions from the DMA port are forwarded as follows:

- If the address hits any of the outbound translation windows, the transaction is forwarded to the PCI port, with the address translated. See Section 11.5.2, “PCI Outbound Address Translation,” for more information.
- All other transactions are forwarded to the CSB port.

11.5.2 PCI Outbound Address Translation

Outbound address translation is provided to allow the outbound transactions to access any address over the PCI memory or I/O space. Translation window base addresses are defined in the PCI outbound base address registers. See Section 11.4.2, “PCI Outbound Base Address Registers (POBARn),” for more information. Transactions to these address ranges are issued on the PCI bus with a translated address. The translation addresses are defined in the associated PCI outbound translation address registers (POTARs).

Figure 11-7 shows an example translation window for outbound memory accesses.

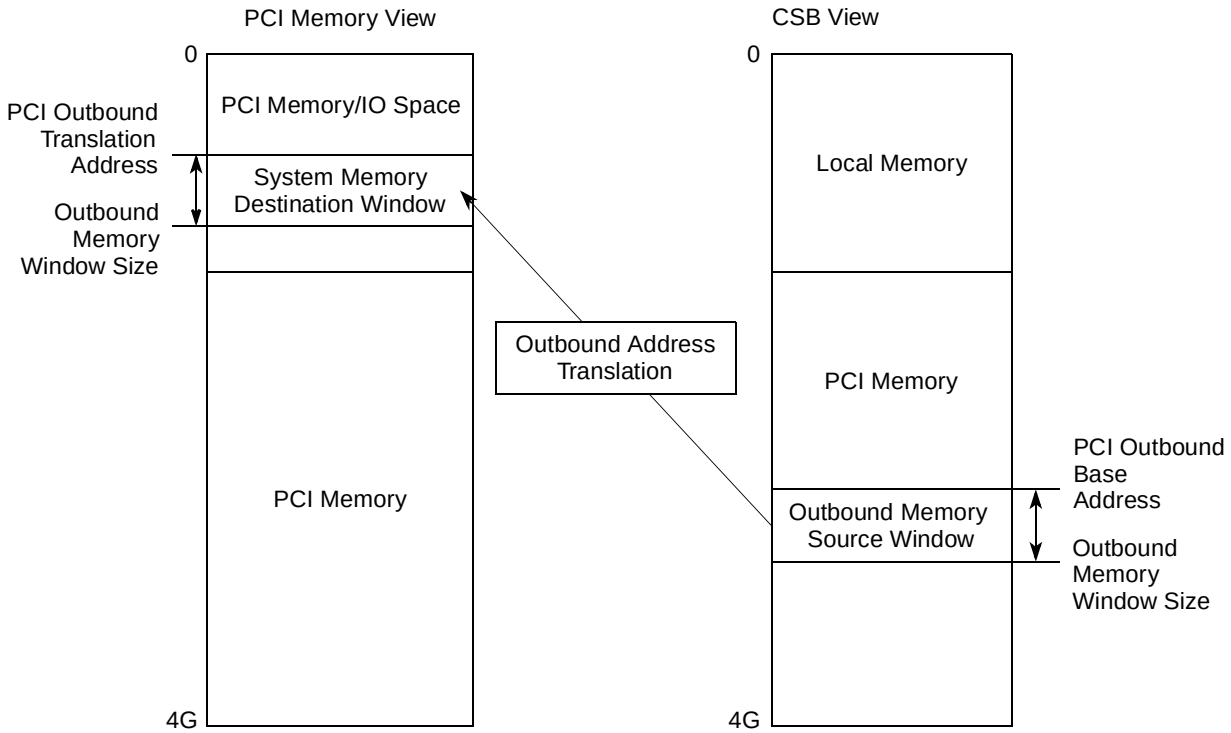


Figure 11-7. Outbound PCI Memory Address Translation

The six sets of outbound translation registers allow six simultaneous translation windows to the PCI port. Software can move and adjust the memory window translations and sizes during run-time. This allows software to access different PCI memory/IO spaces on-the-fly, but the PCI outbound translation source windows must not overlap. However, outbound translation destination windows can be overlapped.

11.5.3 Transaction Ordering

The following rules are applied to maintain proper ordering of transactions:

- The transactions arriving from each port are dispatched to the destination port in the order of arrival. The dispatch order of transactions arriving on different ports is not necessarily maintained.
- A read transaction that originates at the CSB port and reads from the PCI port pulls out of the IOS any posted writes that originated on the PCI port and were posted before the read data arrives from the PCI.
- The IOS can always accept a write from the PCI port without forcing the PCI port to first accept a read.

Chapter 12
DMA/Messaging Unit

The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This unit operates with generic messages and doorbell registers. Figure 12-1 is a block diagram of the DMA/messaging unit.

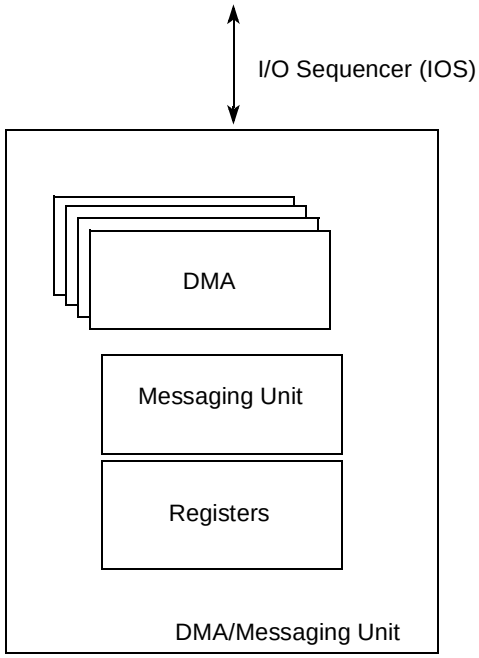


Figure 12-1. DMA/Messaging Unit Block Diagram

This block also provides a DMA controller, which transfers blocks of data independent of the local processor or PCI hosts. The DMA module has four high-speed DMA channels, which share buffer space in the I/O sequencer (IOS) to facilitate the gathering and sending of data.

12.1 Features

The DMA/messaging unit includes the following features:

- Message and doorbell registers for inter-processor communication
- DMA controller
  - — Four DMA channels
  - — Concurrent execution across multiple channels with programmable bandwidth control
  - — Misaligned transfer capability

— Data chaining and direct mode
— Interrupt on completed segment, chain, and error

12.2 Memory Map/Register Definition

Table 12-1 lists the address and access of the memory map module.

Table 12-1. Module Memory Map

Offset	Register	Access	Reset	Section/Page
0x0_8030	OMISR—Outbound message interrupt status register	Mixed	0x0000_0000	12.3.1/12-3
0x0_8034	OMIMR—Outbound message interrupt mask register	R/W	0x0000_0000	12.3.2/12-5
0x0_8050	IMR0—Inbound message register 0	R/W	0x0000_0000	12.3.3/12-6
0x0_8054	IMR1—Inbound message register 1	R/W	0x0000_0000	12.3.3/12-6
0x0_8058	OMR0—Outbound message register 0	R/W	0x0000_0000	12.3.4/12-6
0x0_805C	OMR1—Outbound message register 1	R/W	0x0000_0000	12.3.4/12-6
0x0_8060	ODR—Outbound doorbell register	R/W	0x0000_0000	12.3.5/12-6
0x0_8068	IDR—Inbound doorbell register	R/W	0x0000_0000	12.3.5/12-6
0x0_8080	IMISR—Inbound message interrupt status register	Mixed	0x0000_0000	12.3.6/12-8
0x0_8084	IMIMR—Inbound message interrupt mask register	R/W	0x0000_0000	12.3.7/12-9
0x0_8100	DMAMR0—DMA 0 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8104	DMASR0—DMA 0 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8108	DMACDAR0—DMA 0 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8110	DMASAR0—DMA 0 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8118	DMADAR0—DMA 0 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_8120	DMABCR0—DMA 0 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_8124	DMANDAR0—DMA 0 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_8180	DMAMR1—DMA 1 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8184	DMASR1—DMA 1 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8188	DMACDAR1—DMA 1 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8190	DMASAR1—DMA 1 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8198	DMADAR1—DMA 1 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_81A0	DMABCR1—DMA 1 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_81A4	DMANDAR1—DMA 1 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_8200	DMAMR2—DMA 2 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8204	DMASR2—DMA 2 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8208	DMACDAR2—DMA 2 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8210	DMASAR2—DMA 2 source address register	R/W	0x0000_0000	12.3.8.4/12-14

Table 12-1. Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8218	DMADAR2—DMA 2 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_8220	DMABCR2—DMA 2 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_8224	DMANDAR2—DMA 2 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_8280	DMAMR3—DMA 3 mode register	R/W	0x0000_0000	12.3.8.1/12-10
0x0_8284	DMASR3—DMA 3 status register	R/W	0x0000_0000	12.3.8.2/12-12
0x0_8288	DMACDAR3—DMA 3 current descriptor address register	R/W	0x0000_0000	12.3.8.3/12-13
0x0_8290	DMASAR3—DMA 3 source address register	R/W	0x0000_0000	12.3.8.4/12-14
0x0_8298	DMADAR3—DMA 3 destination address register	R/W	0x0000_0000	12.3.8.5/12-14
0x0_82A0	DMABCR3—DMA 3 byte count register	R/W	0x0000_0000	12.3.8.6/12-15
0x0_82A4	DMANDAR3—DMA 3 next descriptor address register	R/W	0x0000_0000	12.3.8.7/12-15
0x0_82A8	DMAGSR—DMA general status register	R	0x0000_0000	12.3.8.8/12-16
0x0_82B0–0x0_82FF	Reserved	—	—	—

12.3 Register Descriptions

The following sections describe the DMA/messaging unit configuration, control, and status registers.

NOTE

The registers described in this section use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data. No byte swapping occurs when the registers are accessed from the PCI bus.

12.3.1 Outbound Message Interrupt Status Register (OMISR)

OMISR contains the interrupt status of the doorbell and outbound message registers. A PCI device acknowledges the outbound message interrupt by writing a 1 to the appropriate status bit: OMISR[OM1I] or OMISR[OM0I]. Setting one of these bits clears both the interrupt and the corresponding status bit. The local processor provokes an outbound message interrupt by writing to either of the two outbound message registers: OMR0 or OMR1. OMISR can be accessed from the CSB or the PCI bus, but it is normally accessed only from the PCI bus. Figure 12-2 shows the OMISR fields.

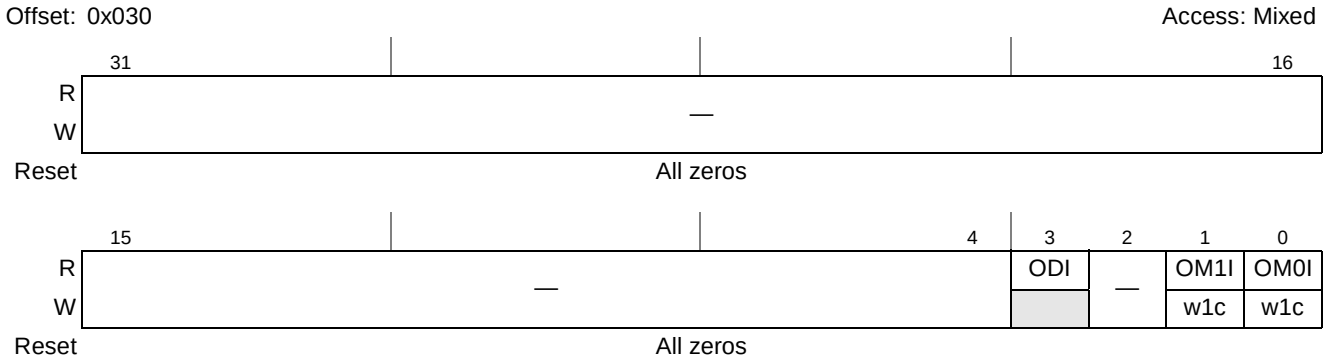


Figure 12-2. Outbound Message Interrupt Status Register (OMISR)

Table 12-2 describes the OMISR register.

Table 12-2. OMISR Field Descriptions

Bits	Name	Description
31–4	—	Reserved
3	ODI	Outbound doorbell interrupt. This read-only bit indicates the status of the ODR bits. It is masked by OMIMR[ODIM]. 0 No outbound doorbell interrupt. 1 There is an outbound doorbell interrupt.
2	—	Reserved
1	OM1I	Outbound message 1 interrupt. When set, indicates that there is an outbound message 1 interrupt. Write 1 to this position to clear this bit. 0 No outbound message 1 interrupt. 1 There is an outbound message 1 interrupt.
0	OM0I	Outbound message 0 interrupt. When set, indicates that there is an outbound message 0 interrupt. Write 1 to this position to clear this bit. 0 No outbound message 0 interrupt. 1 There is an outbound message 0 interrupt.

12.3.2 Outbound Message Interrupt Mask Register (OMIMR)

OMIMR contains the interrupt mask of the doorbell and message register events generated by the local processor. OMIMR can be read from the CSB or the PCI bus, but it can be written only from the PCI bus. Figure 12-3 shows the OMIMR.

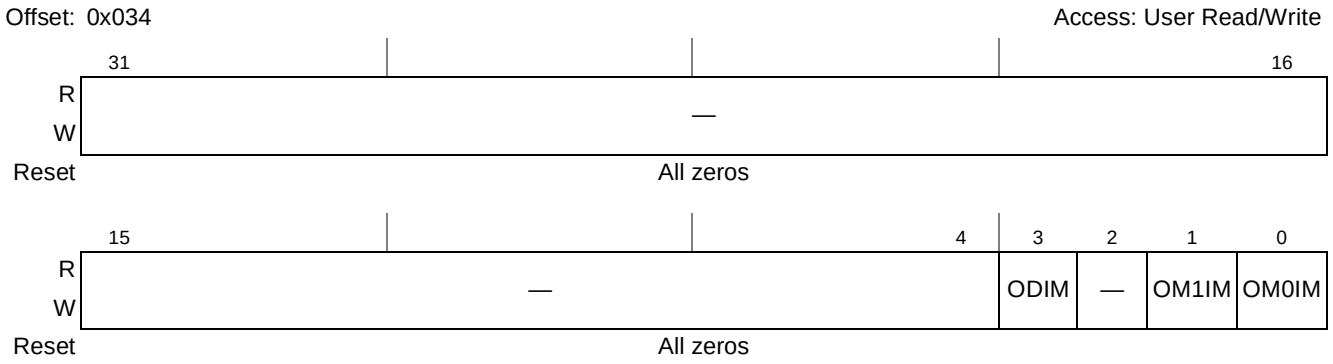


Figure 12-3. Outbound Message Interrupt Mask Register (OMIMR)

Table 12-3 describes the OMIMR register.

Table 12-3. OMIMR Field Descriptions

Bits	Name	Description
31–4	—	Reserved
3	ODIM	Outbound doorbell interrupt mask. 0 Outbound doorbell interrupt is allowed 1 Outbound doorbell interrupt is masked
2	—	Reserved
1	OM1IM	Outbound message 1 interrupt mask. 0 Outbound message 1 interrupt is allowed 1 Outbound message 1 interrupt is masked
0	OM0IM	Outbound message 0 interrupt mask. 0 Outbound message 0 interrupt is allowed 1 Outbound message 0 interrupt is masked

12.3.3 Inbound Message Registers (IMR0–IMR1)

The inbound message registers can be read from the PCI bus and the CSB in both host and agent modes. They can be written only from the PCI bus. Figure 12-4 shows the IMR0 and IMR1 fields.

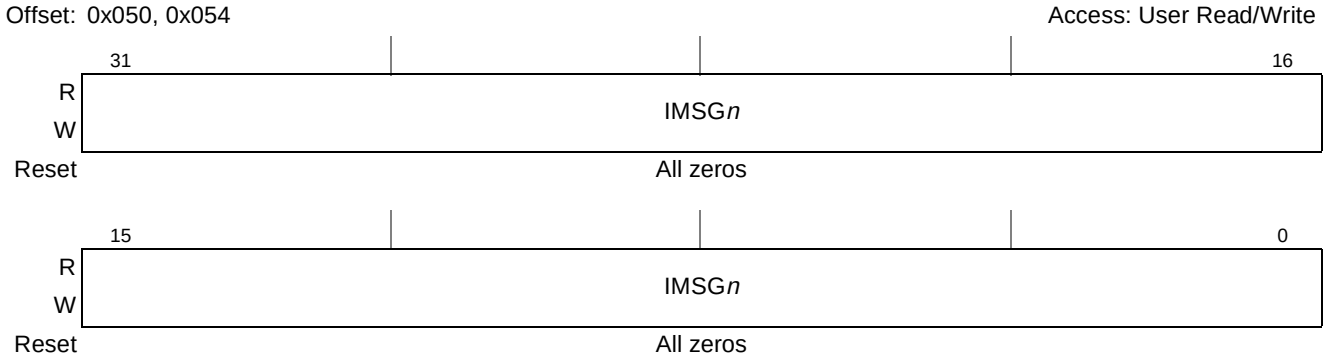


Figure 12-4. Inbound Message Registers (IMR0, IMR1)

Table 12-4 describes the IMR*n* register.

Table 12-4. IMR0 and IMR1 Field Descriptions

Bits	Name	Description
31–0	IMSG*n*	Inbound message *n*. Contains generic data to be passed between the local processor and external hosts.

12.3.4 Outbound Message Registers (OMR0–OMR1)

The outbound message registers can be read from the PCI bus and the CSB in both host and agent modes. They can be written only from the CSB.

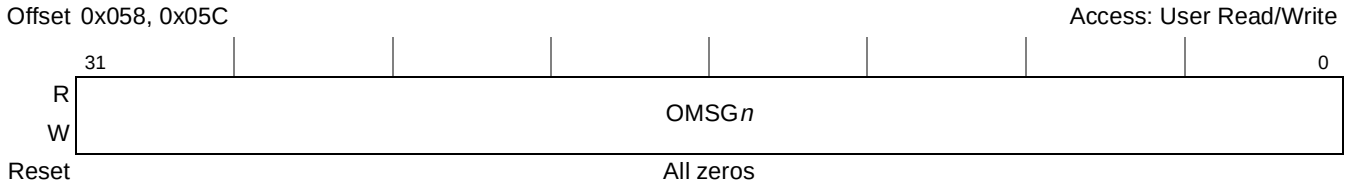


Figure 12-5. Outbound Message Registers (OMR0–OMR1)

Table 12-5 describes the OMR*n* registers.

Table 12-5. OMR0 and OMR1 Field Descriptions

Bits	Name	Description
31–0	OMSG*n*	Outbound message *n*. Contains generic data to be passed between the local processor and external hosts.

12.3.5 Doorbell Registers

The following sections describe the outbound and inbound doorbell registers.

12.3.5.1 Outbound Doorbell Register (ODR)

ODR is accessible from the PCI bus and the CSB in both host and agent modes. Figure 12-6 shows the ODR*n* fields.

Offset: 0x060 Access: User Read/Write

	31–29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	—	ODR 28	ODR 27	ODR 26	ODR 25	ODR 24	ODR 23	ODR 22	ODR 21	ODR 20	ODR 19	ODR 18	ODR 17	ODR 16
Reset	All zeros													

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ODR 15	ODR 14	ODR 13	ODR 12	ODR 11	ODR 10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
Reset	All zeros															

Figure 12-6. Outbound Doorbell Register (ODR)

Table 12-6 describes the ODR registers.

Table 12-6. ODR Field Descriptions

Bits	Name	Description
31–29	—	Reserved
28–0	ODR*n*	Outbound doorbell *n.* Write 1 from the CSB to set. Write 1 from the PCI bus to clear. Writing 0 has no effect. (Writing a bit in this register from the CSB causes an interrupt ($\overline{\text{PCI\_INTA}}$) to be generated.)

12.3.5.2 Inbound Doorbell Register (IDR)

IDR is accessible from the PCI bus and the CSB in both host and agent modes. Figure 12-7 shows the IDR fields.

Offset: 0x068 Access: User Read/Write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMC	IDR30	IDR29	IDR28	IDR27	IDR26	IDR25	IDR24	IDR23	IDR22	IDR21	IDR20	IDR19	IDR18	IDR17	IDR16
Reset	All zeros															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
Reset	All zeros															

Figure 12-7. Inbound Doorbell Register (IDR)

Table 12-7 describes the IDR registers.

Table 12-7. IDR Field Descriptions

Bits	Name	Descriptions
31	IMC	Inbound machine check. Write 1 from the PCI bus to set. Write 1 from the CSB to clear. Writing 0 has no effect. Writing this bit from the PCI bus causes a machine check interrupt to be generated to the local processor.
30–0	IDR*n*	Inbound doorbell *n*. Write 1 from the PCI bus to set. Write 1 from the CSB to clear. Writing 0 has no effect. Writing a bit in this register from the PCI bus causes an interrupt to be generated to the local processor.

12.3.6 Inbound Message Interrupt Status Register (IMISR)

The IMISR contains the interrupt status of the doorbell and message register events. Writing a 1 to IM1I clears the bit. The events are generated by the PCI masters.

Figure 12-8 shows the IMISR fields.

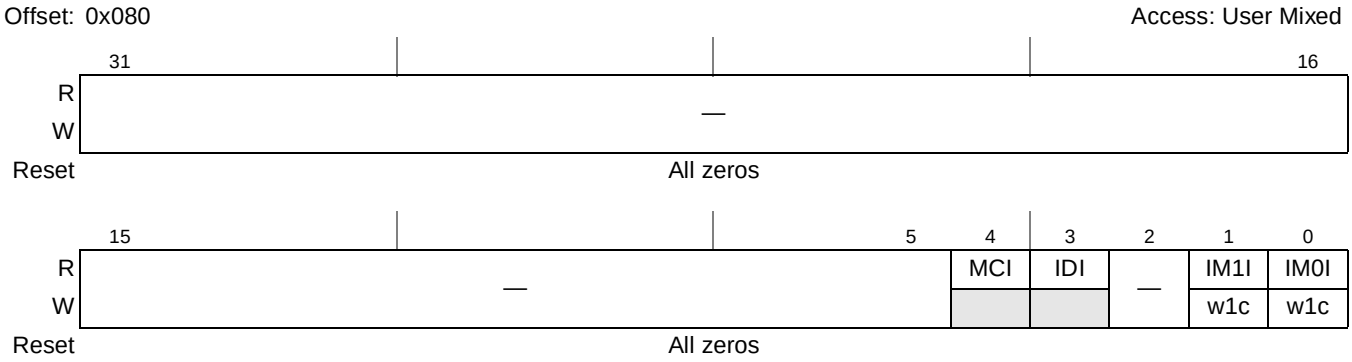


Figure 12-8. Inbound Message Interrupt Status Register (IMISR)

Table 12-8 describes the IMISR register.

Table 12-8. IMISR Field Descriptions

Bits	Name	Descriptions
31–5	—	Reserved
4	MCI	Machine check interrupt. Indicates whether a machine check interrupt condition was generated by setting the IDR[31]. The interrupt is cleared by writing a 1 to IDR[IMC] from the CSB. 0 No machine check interrupt 1 There is a machine check interrupt
3	IDI	Inbound doorbell interrupt. Indicates whether an inbound doorbell interrupt occurred. 0 No inbound doorbell interrupt 1 There is an inbound doorbell interrupt
2	—	Reserved

Table 12-8. IMISR Field Descriptions (continued)

Bits	Name	Descriptions
1	IM1I	Inbound message 1 interrupt. Indicates whether an inbound message 1 interrupt occurred. Write 1 to this position to clear this bit. 0 No inbound message 1 interrupt. 1 There is an inbound message 1 interrupt.
0	IM0I	Inbound message 0 interrupt. Indicates whether an inbound message 0 interrupt occurred. Write 1 to this position to clear this bit. 0 No inbound message 0 interrupt. 1 There is an inbound message 0 interrupt.

12.3.7 Inbound Message Interrupt Mask Register (IMIMR)

This register contains the interrupt mask of the doorbell and message register events generated by the PCI master. Figure 12-9 shows the IMIMR fields.

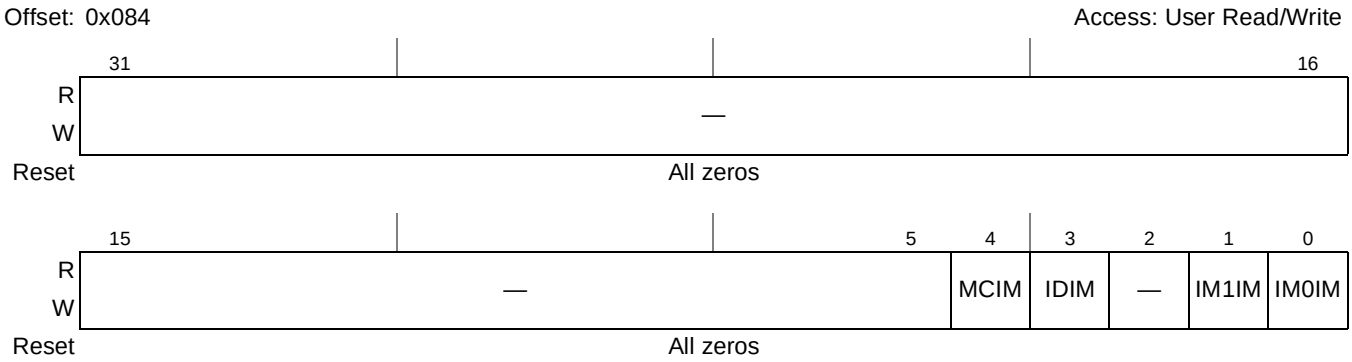


Figure 12-9. Inbound Message Interrupt Mask Register (IMIMR)

Table 12-9 describes the IMISR register.

Table 12-9. IMIMR Field Descriptions

Bits	Name	Description
31–5	—	Reserved
4	MCIM	Machine check interrupt mask. 0 Machine check interrupt from the IDR is allowed 1 Machine check interrupt is masked. IMISR[MC1] is cleared
3	IDIM	Inbound doorbell interrupt mask. 0 Inbound doorbell interrupt is allowed 1 Inbound doorbell interrupt is masked. IMISR[IDI] is cleared.
2	—	Reserved
1	IM1IM	Inbound message 1 interrupt mask. 0 Inbound message 1 interrupt is allowed 1 Inbound message 1 interrupt is masked. IMISR[IM1I] is cleared
0	IM0IM	Inbound message 0 interrupt mask. 0 Inbound message 0 interrupt is allowed 1 Inbound message 0 interrupt is masked. IMISR[IM0I] is cleared

12.3.8 DMA Registers

Each DMA channel has a set of seven 32-bit registers (mode, status, current descriptor address, next descriptor address, source address, destination address, and byte count) to support transactions. The following sections describe the format of the DMA support registers.

12.3.8.1 DMA Mode Register (DMAMR*n*)

This section describes the DMA mode register. The mode register allows software to start the DMA transfer and to control various DMA transfer characteristics. Figure 12-10 shows the DMAMR*n* fields.

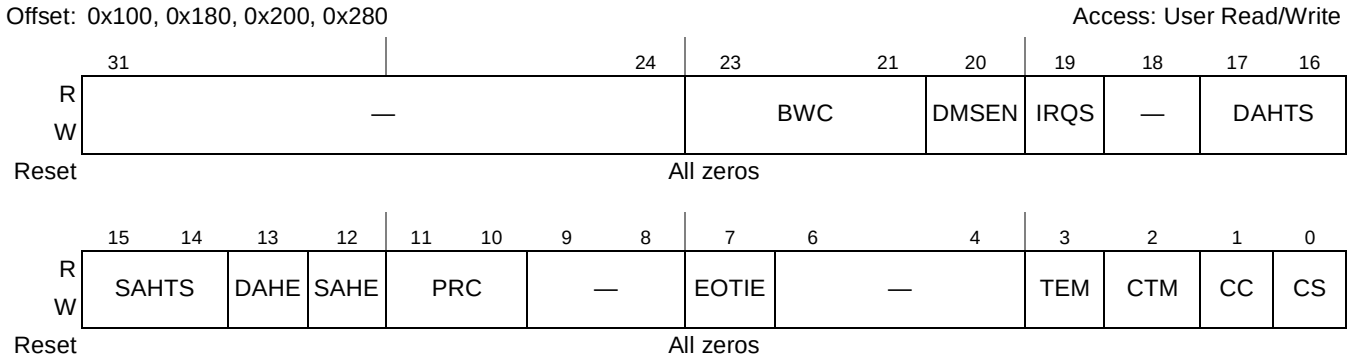


Figure 12-10. DMA Mode Register (DMAMR*n*)

Table 12-10 describes the DMAMR*n* register.

Table 12-10. DMAMR*n* Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23–21	BWC	Bandwidth control. Only applies when multiple channels are executing transfers concurrently. The field determines how many cache lines a given channel is allowed to transfer after it is granted access to the IOS interface and before it releases the interface to the next channel. This allows the user to prioritize the DMA channels. The BWC values are listed as follows: 000 1 cache line 001 2 cache lines 010 4 cache lines 011 8 cache lines 100 16 cache lines Others Reserved
20	DMSEN	Direct mode snoop enable. This bit controls snooping of direct mode DMA transactions. 0 Snooping is disabled 1 Snooping is enabled
19	IRQS	Interrupt steer. This bit determines the destination of the DMA interrupts. 0 All DMA interrupts are routed to the on-chip interrupt controller 1 All DMA interrupts are routed to the PCI bus through $\overline{\text{PCI\_INTA}}$
18	—	Reserved

Table 12-10. DMAMR*n* Field Descriptions (continued)

Bits	Name	Description
17–16	DAHTS	Destination address hold transfer size. This field indicates the transfer size used for each transaction when DAHE is 1. The byte count register must be in multiples of the size, and the destination address register must be aligned based on the size. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
15–14	SAHTS	Source address hold transfer size. This field indicates the transfer size used for each transaction when SAHE is 1. The byte count register must be in multiples of the size, and the source address register must be aligned based on the size. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
13	DAHE	Destination address hold enable. This bit allows the DMA controller to hold the destination address constant for every transfer. The size used for transfer is indicated by DAHTS. Note that hardware supports only aligned transfers for this feature. 0 Do not hold the destination address constant 1 Hold the destination address constant **Note:** The DMA does not support address hold for both the source and the destination at the same transfer.
12	SAHE	Source address hold enable. This bit allows the DMA controller to hold the source address constant for every transfer. The size used for transfer is indicated by SAHTS. Note that hardware supports only aligned transfers for this feature. 0 Do not hold the source address constant 1 Hold the source address constant **Note:** The DMA does not support address hold for both the source and the destination at the same transfer.
11–10	PRC	PCI read command. This field indicates the type of PCI read command to use. 00 Reserved 01 PCI read line 10 PCI read multiple 11 Reserved
9–8	—	Reserved
7	EOTIE	End-of-transfer interrupt enable. This bit determines whether an interrupt is generated at the completion of a DMA transfer. End-of-transfer is defined as the end of a direct mode transfer or in chaining mode, as the end of the transfer of the last segment of a chain. 0 No EOT interrupt is generated 1 EOT interrupt is generated
6–4	—	Reserved
3	TEM	Transfer error mask. This bit determines the DMA response in the event of a transfer error. 0 The DMA will halt when a transfer error occurs. 1 The DMA will complete the transfer regardless of whether a transfer error occurs. **Note:** Regardless of the setting of TEM, if an error condition was detected during the DMA transfer, it will cause DMASR*n*[TE] to be set.
2	CTM	Channel transfer mode. 0 Chaining mode 1 Direct mode

Table 12-10. DMAMR*n* Field Descriptions (continued)

Bits	Name	Description
1	CC	Channel continue. This bit applies only to chaining mode. Setting this bit indicates that the current descriptor segment should be repeated. CC is cleared by the DMA once the repeat takes effect, so it only causes a single repeat. 0 Normal chaining 1 DMACDAR is not loaded from DMANDAR, causing a repeat of the current descriptor segment
0	CS	Channel start. A 0-to-1 transition occurring on this bit when the channel is not busy (SR[CB] bit is 0) will start the DMA process. If the channel is busy and a 0-to-1 transition occurs, the DMA channel will restart from a previous halt condition. A 1-to-0 transition when the channel is busy (CB bit is 1) will halt the DMA process. Nothing happens if the channel is not busy and a 1-to-0 transition occurs. This bit is cleared by the DMA at the end of a transfer.

12.3.8.2 DMA Status Register (DMASR*n*)

This section describes the DMA status register. The status register reports various DMA conditions during and after the DMA transfer. Writing a 1 to a specific set bit clears the bit. Figure 12-11 shows the DMASR*n* fields.

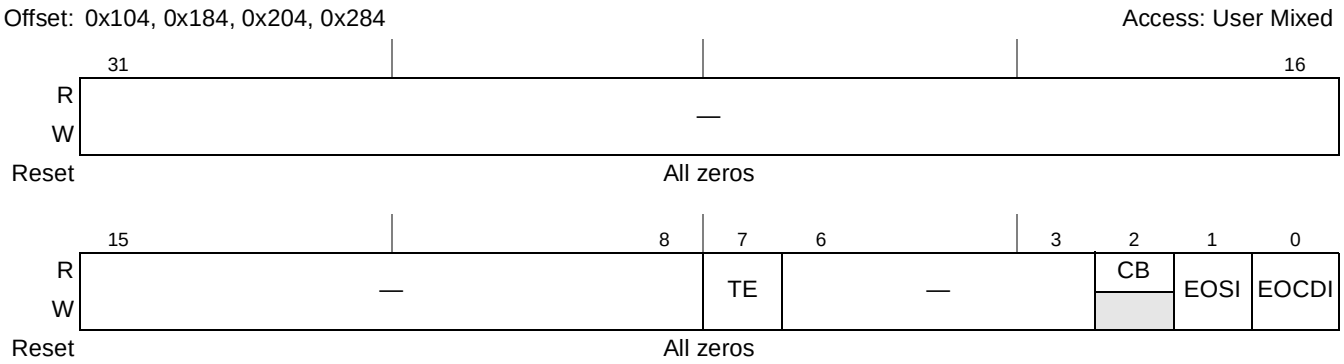


Figure 12-11. DMA Status Register (DMASR*n*)

Table 12-11 describes the DMASR*n* register.

Table 12-11. DMASR*n* Field Descriptions

Bits	Name	Description
31–8	—	Reserved
7	TE	Transfer error. Set when there is an error condition during the DMA transfer.
6–3	—	Reserved
2	CB	Channel busy. This bit indicates whether the channel is busy. It is cleared as a result of any of the following conditions: an error or completion of the DMA transfer. 0 No DMA transfer is currently in progress 1 A DMA transfer is currently in progress

Table 12-11. DMASR*n* Field Descriptions (continued)

Bits	Name	Description
1	EOSI	End-of-segment interrupt. After transferring a segment of data, if the DMACDAR*n*[EOSIE] bit in the current descriptor address register is set, this bit is set and an interrupt is generated.
0	EOCDI	End-of-chain/direct interrupt. When the last DMA transfer is finished, either in chaining or direct mode, if DMAMR[EOTIE] is set, this bit is set and an interrupt is generated.

12.3.8.3 DMA Current Descriptor Address Register (DMACDAR*n*)

DMACDAR*n* contains the address of the current segment descriptor being transferred. In chaining mode, software must initialize this register to point to the first descriptor in the chain. After processing the first descriptor, the DMA controller moves the contents of the next descriptor address register into DMACDAR, loads the following descriptor into DMANDAR, and executes the current transfer. Figure 12-12 shows the DMACDAR*n* fields.

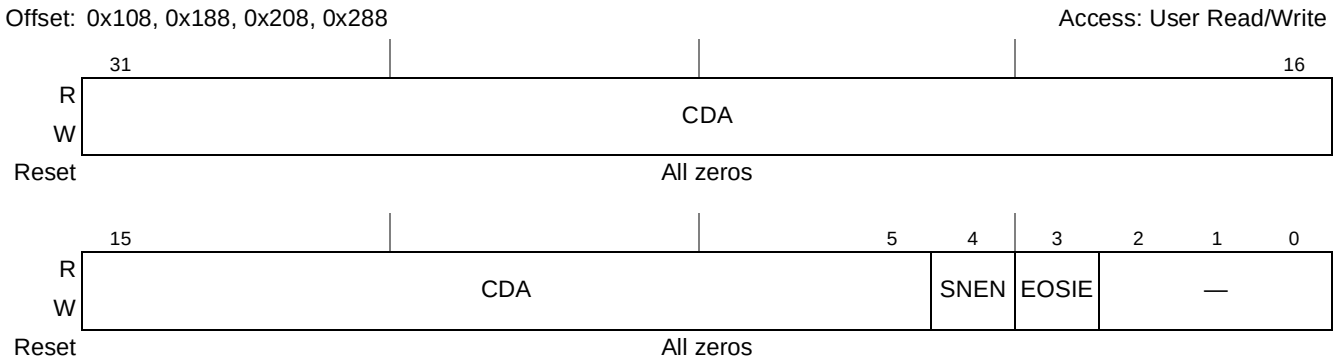


Figure 12-12. DMA Current Descriptor Address Register (DMACDAR*n*)

Table 12-12 describes the DMACDAR*n* register.

Table 12-12. DMACDAR*n* Field Descriptions

Bits	Name	Description
31–5	CDA	Current descriptor address. This field contains the current descriptor address of the segment descriptor in memory. It must be aligned on an 8-word boundary.
4	SNEN	Snoop enable. 0 Snooping is disabled on DMA transactions of the current segment. 1 Snooping is enabled on DMA transactions of the current segment.
3	EOSIE	End-of-segment interrupt enable 0 No end-of-segment interrupt is generated. 1 An interrupt is generated when the current DMA transfer for the current descriptor is finished.
2–0	—	Reserved

12.3.8.4 DMA Source Address Register (DMASAR*n*)

DMASAR*n* indicates the address from which the DMA controller is reading data. The software must ensure that this is a valid memory address. Figure 12-13 shows the DMASAR*n*.

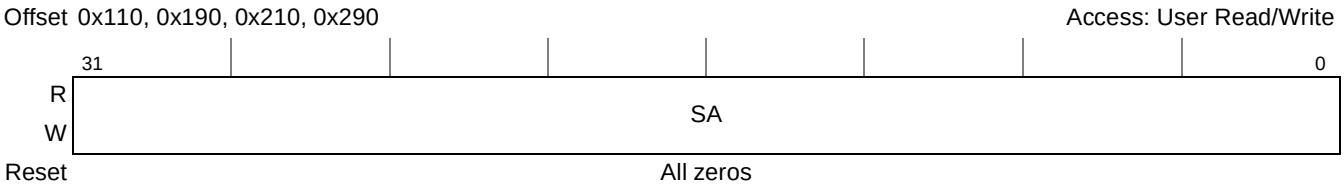


Figure 12-13. DMA Source Address Register (DMASAR*n*)

Table 12-13 describes the DMASAR*n* register.

Table 12-13. DMASAR*n* Field Descriptions

Bits	Name	Description
31–0	SA	Source address of DMA transfer. The content of this field is updated after each DMA read operation.

12.3.8.5 DMA Destination Address Register (DMADAR*n*)

DMADAR*n* indicates the address to which the DMA controller is writing data. The software must ensure that this is a valid memory address. Figure 12-14 shows the DMADAR*n* fields.

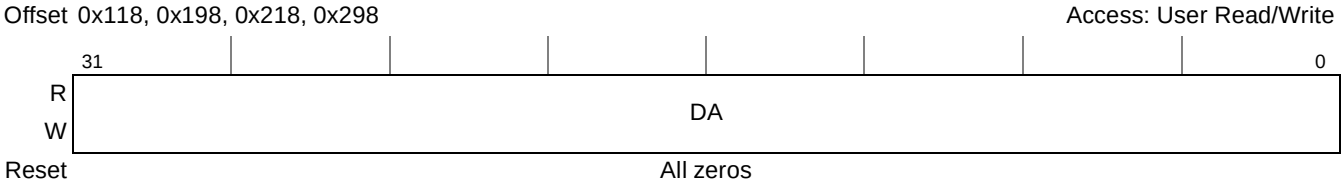


Figure 12-14. DMA Destination Address Register (DMADAR*n*)

Table 12-14 describes the DMADAR*n* register.

Table 12-14. DMASAR*n* Field Descriptions

Bits	Name	Description
31–0	DA	Destination address of DMA transfer.Updated after each DMA write operation.

12.3.8.6 DMA Byte Count Register (DMABCR*n*)

DMABCR*n* contains the number of bytes per transfer (maximum transfer size is 64 Mbytes). Figure 12-15 shows the DMABCR*n*.

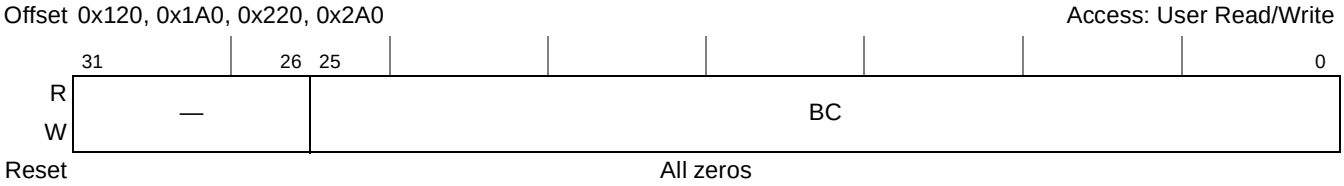


Figure 12-15. DMA Byte Count Register (DMABCR*n*)

Table 12-15 describes the DMABCR*n* register.

Table 12-15. DMABCR*n* Field Descriptions

Bits	Name	Description
31–26	—	Reserved
25–0	BC	Byte count. This field contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. Maximum transfer size is 64 Mbytes.

12.3.8.7 DMA Next Descriptor Address Register (DMANDAR*n*)

DMANDAR*n* contains the address for the next segment descriptor in the chain. In chaining mode, this register is loaded from the ‘next descriptor’ field of the descriptor to which the current descriptor address register is pointing. Figure 12-16 shows the DMANDAR*n*.

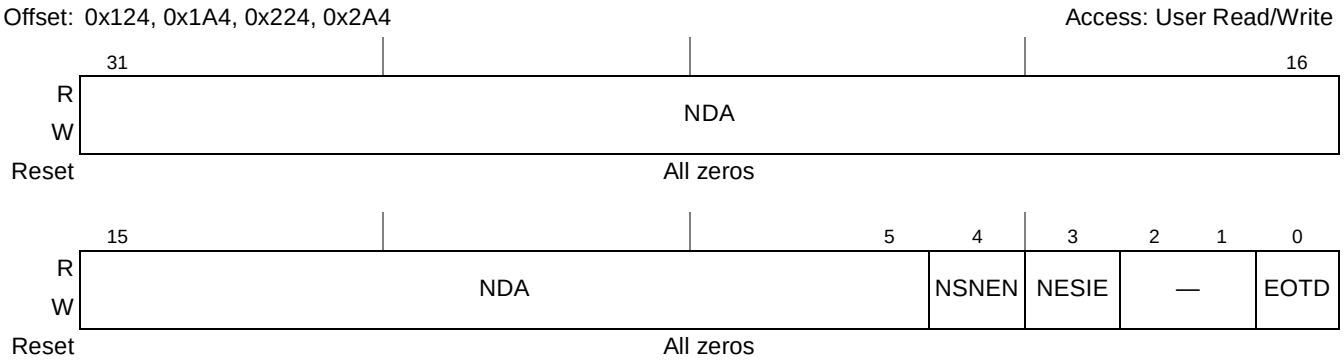


Figure 12-16. DMA Next Descriptor Address Register (DMANDAR*n*)

Table 12-16 describes the DMANDAR*n* register.

Table 12-16. DMANDAR*n* Field Descriptions

Bits	Name	Descriptions
31–5	NDA	Next descriptor address. This field contains the next descriptor address of the next segment descriptor in memory. It must be aligned on an 8-word boundary.
4	NSNEN	Next snoop enable. 0 Snooping is disabled on DMA transactions. 1 Snooping is enabled on DMA transactions.
3	NEOSIE	Next end-of-segment interrupt enable. 0 No end-of-segment interrupt is generated. 1 An interrupt is generated when the DMA transfer for the next descriptor is finished.
2–1	—	Reserved
0	EOTD	End-of-transfer descriptor. 0 This descriptor contains a link to another descriptor. 1 This descriptor is the last to be executed.

12.3.8.8 DMA General Status Register (DMAGSR)

DMAGSR provides faster access to the status bits by combining the status bits of all of the DMA channels into one register. Each byte of this register provides the value of bits 7–0 of a channel's DMA status register. These bits are cleared by writing to the individual DMA status registers. Figure 12-17 shows the DMAGSR fields.

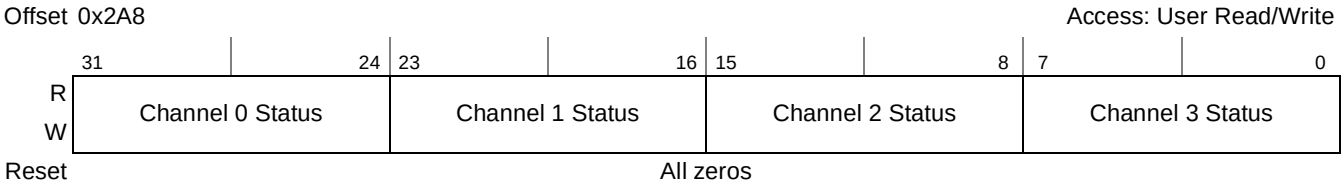


Figure 12-17. DMA General Status Register (DMAGSR)

12.4 Functional Description

12.4.1 Message Unit

An embedded processor is often part of a larger system containing many processors and distributed memory. These processors tend to work on tasks independent of the host and other peripheral processors in the system. Because of the independent nature of the tasks, it is necessary to provide a communication mechanism between the peripheral processors and the rest of the system. One such method is the use of messages. This block provides a messaging unit to further facilitate communications between host and peripheral. The message unit uses generic messages and doorbell registers.

12.4.1.1 Messaging Registers (IMR0–IMR1, OMR0–OMR1)

There are two 32-bit inbound message registers (IMR0–IMR1) and two 32-bit outbound message registers (OMR0–OMR1). IMR0 and IMR1 allow a remote host or PCI master to write a 32-bit value that, in turn,

causes an interrupt request to the on-chip interrupt controller that drives an interrupt line to the local processor. OMR0 and OMR1 allow the local processor to write an outbound message which, in turn, causes the outbound interrupt signal $\overline{\text{PCI\_INTA}}$ to assert.

The interrupt to the local processor is cleared by writing 1 to the appropriate IMISR bit. The interrupt to PCI ($\overline{\text{PCI\_INTA}}$) is cleared by writing 1 to the appropriate OMISR bit.

12.4.1.2 Doorbell Registers (IDR and ODR)

This block contains the inbound doorbell register (IDR) and the outbound doorbell register (ODR). The inbound doorbell allows a remote processor to set a bit in the register from the PCI bus. This, in turn, generates an interrupt request to the on-chip interrupt controller that drives an interrupt line to the local processor. The local processor can write to the ODR, which causes the outbound interrupt signal $\overline{\text{PCI\_INTA}}$ to assert, thus interrupting the remote processor on the PCI bus.

The interrupt to the local processor is cleared by writing 1 to the appropriate IDR bit. The interrupt to PCI ($\overline{\text{PCI\_INTA}}$) is cleared by writing 1 to the appropriate ODR bit.

12.4.2 DMA Controller

The DMA controller transfers blocks of data independent of the local processor or PCI hosts. Data movement occurs on the PCI bus and/or CSB. The DMA module has four high-speed DMA channels, which share buffer space in the IOS to facilitate the gathering and sending of data. Both the local processor and PCI masters can initiate a DMA transfer.

Features of the DMA controller include the following:

- Four channels
- Concurrent execution across multiple channels with programmable bandwidth control
- All channels are accessible by local processor and remote PCI masters
- Unaligned transfer capability
- Data chaining and direct mode
- Interrupt on completed segment, chain, and error

Figure 12-18 shows a diagram of the DMA controller in the integrated device.

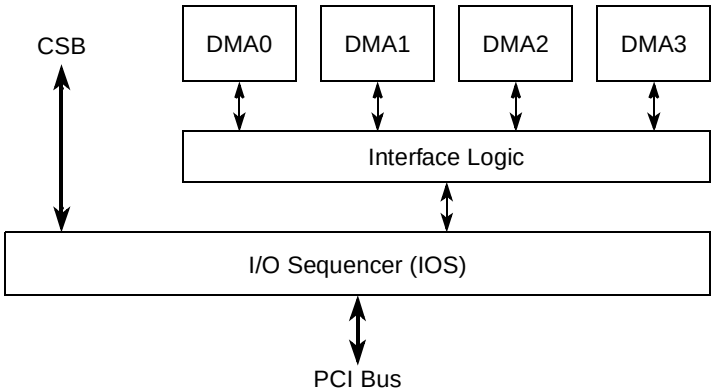


Figure 12-18. DMA Controller Block Diagram

12.4.3 DMA Operation

The DMA controller operates in the following two modes:

- Direct mode, in direct mode, the DMA controller does not read a chain of descriptors from memory but instead uses the current parameters in the DMA registers to start a DMA transfer. The DMA transfer finishes after all the bytes specified in the byte count register have been transferred. See Section 12.5.1, “Initialization Steps in Direct Mode,” for more details on initialization steps.
- Chaining mode, in chaining mode, the DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the descriptor information loaded for each segment. Once the current segment is finished, the DMA controller reads the next descriptor from memory and begins another DMA transfer. The process is finished if the current descriptor is the last one in the chain. See Section 12.5.2, “Initialization Steps in Chaining Mode,” for more details on initialization steps.

In both modes, setting the start bit in the DMA mode register begins the DMA transfer.

The DMA controller supports unaligned transfers for both the source and destination addresses. It gathers data beginning at the source address and aligns the data accordingly before sending it to the destination address. The DMA controller assumes that the source and destination addresses are valid PCI or CSB memory addresses.

Accesses to CSB memory depend on the alignment of the source and destination addresses and the size of the transfer. The DMA controller transfers a full cache line whenever possible. Misaligned destination addresses result in sub-transfers of less than a cache line on the initial and final beats of the transfer; intermediate beats transfer full cache lines. Configuring a DMA channel for address hold mode DMAMR*n* precludes cache line transfers.

PCI memory read operations depend on the PRC (PCI read command) field in the mode register, the alignment of the source address, and the size of the transfer. The DMA controller attempts to read a full cache line whenever possible. Writing to PCI memory depends on the alignment of the destination address and the size of the transfer.

12.4.3.1 DMA Coherency

The four DMA channels use up to four cache lines (128 bytes) of buffer space in the IOS in addition to 16 bytes of local buffer space. Because no address snooping occurs in these internal queues, data posted in these queues is not visible to the rest of the system while a DMA transfer is in progress. It is the responsibility of application software to ensure the coherency of the region being transferred during the DMA process.

Snooping of the CPU or processor data cache is selectable during DMA transactions. A snoop bit is provided in the DMA current descriptor address register (DMACDAR*n*) and the DMA next descriptor address register (DMANDAR*n*) that allows software to control when the cache is snooped on a per segment basis.

12.4.3.2 Halt and Error Conditions

DMA transfers are halted either by clearing the CS (channel start) bit in the DMA mode register (DMAMR*n*) or when encountering an error condition. In either case, the application software can do one of the following:

- Continue the DMA transfer
- Reconfigure the DMA for a new transfer
- Leave the channel in the halted state

When a DMA channel is halted, its programming model is completely accessible. If the DMA is halted due to an error condition, the TE (transfer error) bit in the DMA status register (DMASR*n*) must be cleared before the transfer can be resumed or a new transfer initiated. Note that the TE bit is not cleared automatically by hardware.

12.4.4 DMA Segment Descriptors

DMA segment descriptors contain the source and destination addresses of the data segment, the segment byte count, and a link to the next descriptor. Segment descriptors are built on cache-line (32-byte) boundaries in either CSB or PCI memory and are linked together into chains using the next-descriptor-address field.

Table 12-17. DMA Segment Descriptor Fields

Descriptor Field	Description
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA source address register (DMASAR*n*).
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA destination address register (DMADAR*n*).
Next descriptor address	Points to the next descriptor in memory. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA next descriptor address register (DMANDAR*n*).
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA byte count register (DMABCR*n*).

Application software initializes the current DMA current descriptor address register (DMACDAR*n*) to point to the first descriptor in the chain. For each descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by the descriptor. The DMA controller traverses the descriptor chain until reaching the last descriptor (with its EOTD bit set).

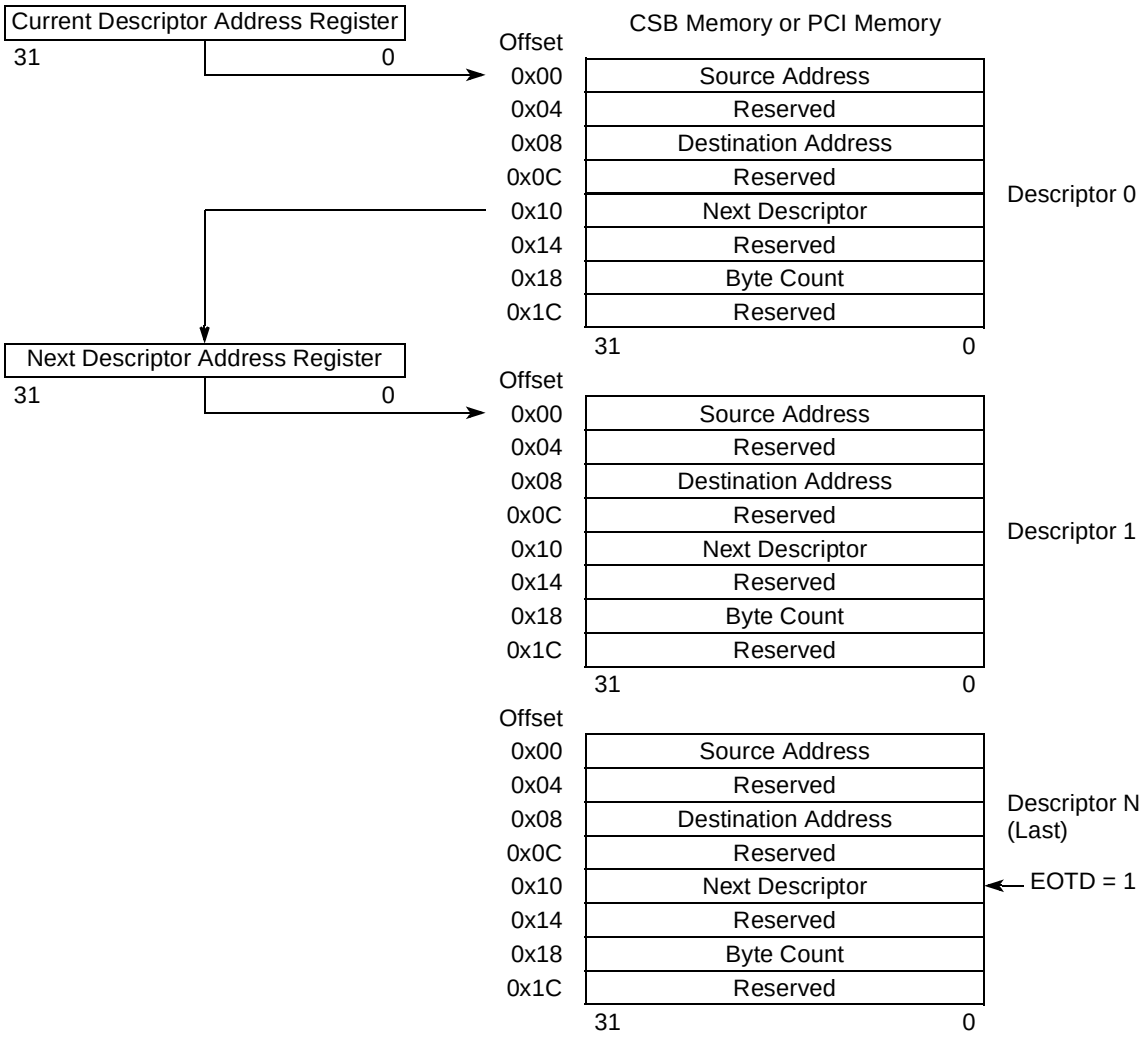


Figure 12-19. DMA Chain of Segment Descriptors

12.4.4.1 Descriptor in Big-Endian Mode

In big-endian mode, the descriptor in CSB memory should be programmed such that data appears in ascending significant-byte order. If segment descriptors are written to memory located in the CSB, they should be treated like they are translated from big-endian to little-endian mode.

Example: Big-endian mode descriptor's data structure. Note that the descriptor structure must be aligned on an 8-word boundary.

```
struct {
          double a;           /* 0x1122334455667788 double word*/
          double b;           /* 0x55667788aabbccdd double word*/
          double c;           /* 0x8765432101234567 double word */
          double d;           /* 0x0123456789abcdef double word */
} Descriptor;
Results:  Source Address = 0x44332211 <MSB..LSB>
          Destination Address = 0x88776655 <MSB..LSB>
          Next Descriptor Address = 0x21436587 <MSB..LSB>
          Byte Count = 0x67452301 <MSB..LSB>
```

12.4.4.2 Descriptor in Little-Endian Mode

In little-endian mode, each segment descriptor should be programmed in descending significant-byte order.

Example: Little-endian mode descriptor's data structure. Note that the descriptor structure must be aligned on an 8-word boundary.

```
struct {
          double a;           /* 0x8877665544332211 double word*/
          double b;           /* 0x1122334488776655 double word*/
          double c;           /* 0x7654321012345678 double word */
          double d;           /* 0x0123456776543210 double word */
} Descriptor;
Results:  Source Address = 0x44332211 <MSB..LSB>
          Destination Address = 0x88776655 <MSB..LSB>
          Next Descriptor Address = 0x12345678 <MSB..LSB>
          Byte Count = 0x76543210 <MSB..LSB>
```

12.5 Initialization/Application Information

12.5.1 Initialization Steps in Direct Mode

The initialization steps of a DMA transfer in direct mode are described as follows:

1. Poll the CB (channel busy) bit in the DMA status register (DMASR*n*) to make sure the DMA channel is idle.
2. Initialize the DMASAR*n*, DMADAR*n*, and the DMABCR*n*.
3. Initialize DMAMR*n*[CTM]) to indicate direct mode. Other control parameters in the mode register can also be initialized here if necessary.
4. First clear then set the DMAMR*n*[CS] to start the DMA transfer.

12.5.2 Initialization Steps in Chaining Mode

The initialization steps of a DMA transfer in chaining mode are described as follows:

1. Build a chain of descriptor segments in memory. Refer to Section 12.4.4, "DMA Segment Descriptors."

2. Poll the DMASR*n*[CB] to make sure the DMA channel is idle.
3. Initialize the DMACDAR*n* to point to the first descriptor in the chain.
4. Initialize the DMAMR*n*[CTM] to indicate chaining mode. Other control parameters in the mode register can also be initialized here if necessary.
5. First clear then set the DMAMR*n*[CS] to start the DMA transfer.

Chapter 13
PCI Bus Interface

The PCI interface is compatible with the *PCI Local Bus Specification*, Rev. 2.3. It is beyond the scope of this manual to document the intricacies of PCI. This chapter describes the PCI controller and provides a basic description of the PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification. Designers of systems incorporating PCI devices should refer to the respective specifications for a thorough description of the PCI buses.

NOTE

Much of the available PCI literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Because this is inconsistent with the terminology in this manual, the terms ‘word’ and ‘double word’ are not used in this chapter. Instead, the number of bits or bytes indicates the exact quantity.

13.1 Introduction

The PCI controller acts as a bridge between the PCI interface and the CSB. The I/O sequencer buffers the data. Figure 13-1 is a high-level block diagram of the PCI controller.

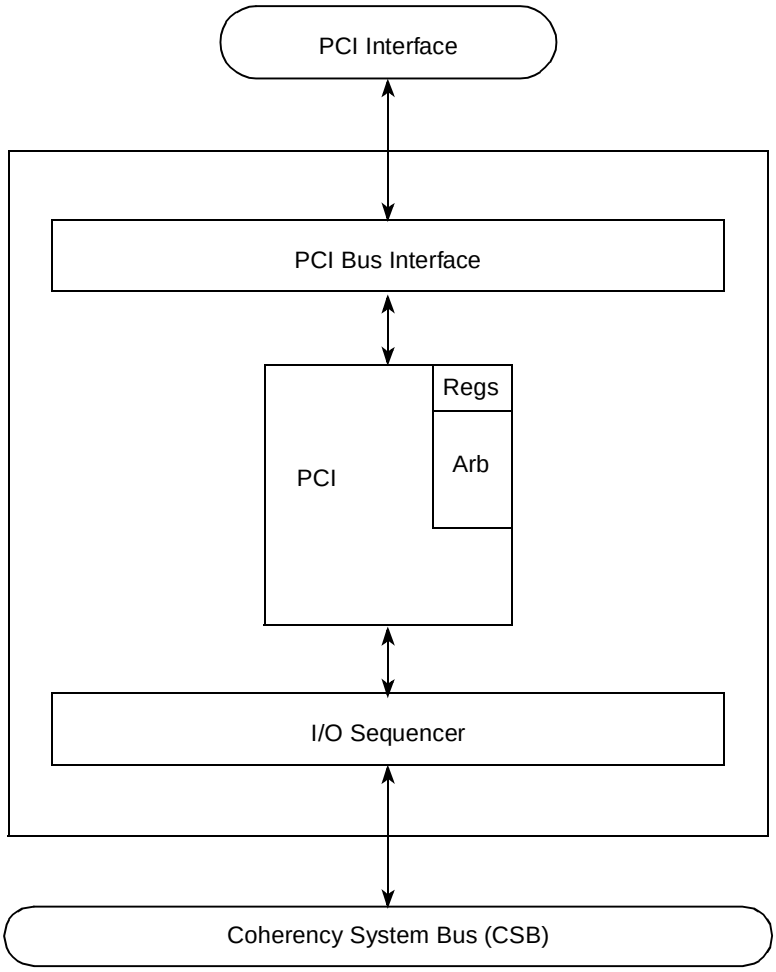


Figure 13-1. PCI Controller Block Diagram

The PCI controller connects the processor and memory system to the I/O components through the PCI system bus. This interface acts as both initiator (master) and target (slave) device. The PCI controller uses a 32-bit multiplexed, address/data bus that can run at frequencies up to 66-MHz. The interface provides address and data parity with error checking and reporting. The interface provides for three physical address spaces—64-bit address memory, 32-bit address I/O, and PCI configuration space.

Note that PCI supports up to three external masters.

The PCI interface can function as either a PCI host bridge referred to as host mode or a peripheral device on the PCI bus referred to as agent mode. See Section 13.4.4.4, “Host Mode Configuration Access,” for more information. Note that the PCI controller can be configured from the PCI bus while in agent mode. An address translation mechanism is provided to map PCI memory windows between the PCI bus and the internal bus.

The PCI interface does not flush pending outbound writes as a result of an inbound read command. Systems must not rely on inbound reads to ensure all pending outbound writes have completed. For example, consider the case where a core writes data to a PCI device and then updates a flag in the local DDR memory indicating the write to PCI has completed. An external PCI master may misread the flag ahead of the actual write transaction's completion on the PCI bus.

13.1.1 Features

The PCI controller includes the following features:

- PCI specification revision 2.3 compliant
- 32-bit PCI interface support
- Host and agent mode support
- Supports accesses to all PCI address spaces
- 64-bit dual-address cycle (DAC) support (as a target only)
- Internal configuration registers accessible from PCI
- On-chip arbitration supporting three masters on PCI
- Arbiter supports two-level priority request/grant signal pairs
- Supports PCI-to-memory and memory-to-PCI streaming
- Memory prefetching of PCI read accesses and support for delayed read transactions
- Supports posting of processor-to-PCI and PCI-to-memory writes
- Supports selectable snooping for inbound transactions
- Address translation units for address mapping between host and peripheral
- Supports parity
- PCI 3.3-V compatible

13.1.2 Modes of Operation

PCI controller modes of operation are determined at reset by the reset configuration word high (RCWH) as described in Section 4.3.2, “Reset Configuration Words.” Table 13-1 summarizes these modes.

Table 13-1. PCI Controller Modes

Parameter	Description	Section/Page
Host/agent configuration	Selects between host and agent mode for the PCI interface.	4.3.2.2.1/4-15
PCI arbiter enable	Enables the on-chip PCI bus arbiter	4.3.2.2/4-14

13.1.2.1 Host/Agent Mode Configuration

The PCI controller can function as either a PCI host bridge (referred to as host mode) or a peripheral device on the PCI bus (referred to as agent mode). Note that host/agent mode selection is determined at power-up as summarized inSection 4.3.2.2.1, “PCI Host/Agent Configuration.”

When the device powers up in host m ode, all inbound configuration accesses are ignored (and thus master aborted). When the device powers up in agent mode, it acknowledges inbound configuration accesses. Note that in PCI agent mode, the PCI controller ignores all PCI memory accesses except those to the memory-mapped registers until inbound address translation is enabled. In agent mode, configuration cycles are acknowledged if CFG_LOCK is 0 (see Section 13.3.3.24, “PCI Function Configuration Register”), either from reset configuration or after being cleared by software.

13.1.2.2 PCI Clock Output Enable Configuration

Normally when the device is in host mode, PCI clock outputs are provided for connection to other agents in the system. However it is possible to disable the PCI clock outputs of the device. When PCI clock outputs are disabled, PCI clock distribution should be done on the board by an external device. The benefit of disabling the PCI clock outputs is that the PCI clock device pins can be used for alternate functions.

13.1.2.3 PCI Arbiter Configuration

The interface can be configured to use an on-chip or off-chip PCI arbiter. Arbitration for PCI is determined by the value in RCWH[PCIARB]. SeeSection 4.3.2.2, “Reset Configuration Word High Register (RCWHR),” for more information.

13.2 External Signal Description

Table 13-2 shows the properties of the PCI signals.

Table 13-2. Signal Properties

Name	Function	Reset State	Pull Up
CPCI_HS_ENUM	CompactPCI hot swap enumerator	High impedance	Required
CPCI_HS_ES	CompactPCI hot swap ejector switch	—	—
CPCI_HS_LED	CompactPCI hot swap LED	Asserted	—
M66EN	66-MHz enable	—	—
PCI_AD[31:0]	PCI*n* address / data	High impedance	—
PCI_C/$\overline{\text{BE}}$[3:0]	PCI bus command / byte enable	High impedance	—
$\overline{\text{PCI\_DEVSEL}}$	PCI device select	High impedance	Required
$\overline{\text{PCI\_FRAME}}$	PCI cycle frame	High impedance	Required
$\overline{\text{PCI\_REQ}}$[0:2]	PCI arbiter requests	Configuration-dependent	Required on inputs
$\overline{\text{PCI\_GNT}}$[0:2]	PCI arbiter grants	Configuration-dependent	—
PCI_IDSEL	PCI initialization device select	—	—
$\overline{\text{PCI\_INTA}}$	PCI interrupt A	High impedance	Required
$\overline{\text{PCI\_IRDY}}$	PCI initiator ready	High impedance	Required
PCI_PAR	PCI parity	High impedance	—
$\overline{\text{PCI\_PERR}}$	PCI parity error	High impedance	Required
$\overline{\text{PCI\_RESET\_OUT}}$	PCI reset output	Asserted	

Table 13-2. Signal Properties (continued)

Name	Function	Reset State	Pull Up
PCI_SERR	PCI system error	High impedance	Required
PCI_STOP	PCI stop	High impedance	Required
PCI_TRDY	PCI target ready	High impedance	Required

Figure 13-2 shows the external PCI signals.

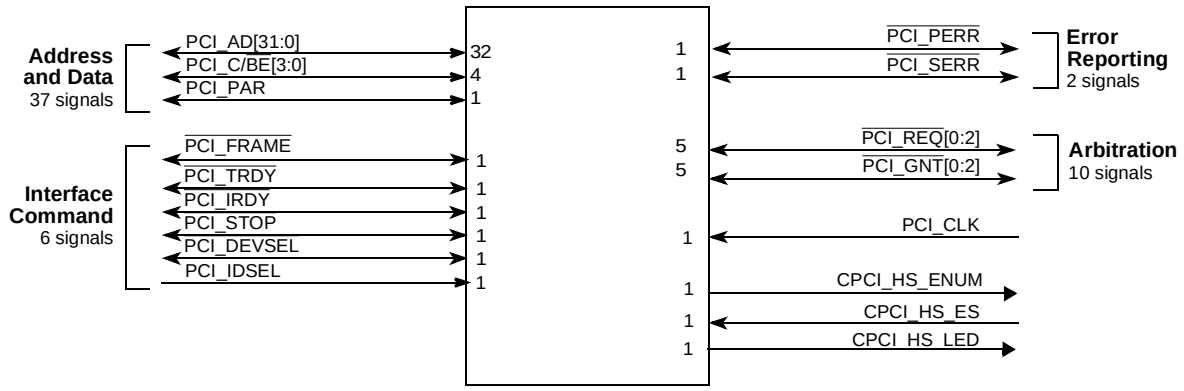


Figure 13-2. PCI Interface External Signals

Table 13-3 contains detailed descriptions of the external PCI interface signals.

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
CPCI_HS_ENUM	O	CompactPCI hot swap enumerator. Used for the hot swap interface to connect to the host as the enumeration request in a compact PCI system. This signal is used for agent mode only.	
		State Meaning	Asserted—This card was inserted and needs to be configured, or this card is about to be extracted and needs to be removed from the system resources list. Negated—No action is needed.
		Timing	Assertion/Negation—No timing is specified
CPCI_HS_ES	I	CompactPCI hot swap ejector switch. Used for agent mode only. In a compact PCI system this input signal is used for the hot swap interface to connect to the ejector switch logic.	
		State Meaning	Asserted—The switch is open. Negated—The switch is closed.
		Timing	Assertion/Negation—No timing is specified
CPCI_HS_LED	O	CompactPCI hot swap LED. Used for the hot swap interface to connect to the hot swap LED in a CompactPCI system. This signal is used for agent mode only.	
		State Meaning	Asserted—Output is driving logic 1 to illuminate the hot swap LED. Negated—Output is driving logic 0 to turn off the hot swap LED.
		Timing	Assertion/Negation—No timing is specified

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
M66EN	I	66-MHz enable. Determines the AC timing of the PCI interface.	
		State Meaning	Asserted—The PCI interface signals use the 66-MHz PCI AC timing parameters. Negated—The PCI interface signals use the 33-MHz PCI AC timing parameters.
		Timing	Assertion/Negation—Constant
PCI_AD[31:0]	I/O	PCI address/data bus. During an address phase, these signals contain a physical address. During a data phase, these signals contain the data bytes.	
	O	Outputs for the bi-directional PCI address/data bus.	
		State Meaning	Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, the PCI address/data bus contain the data being written. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional PCI address/data bus.	
		State Meaning	Asserted/Negated—Represents the address to be decoded as a check for device select during the address phase of a PCI transaction or the data being received during the data phase(s) of a PCI transaction. PCI_AD[7:0] define the LSB and, PCI_AD[31:24] define the MSB.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_C/$\overline{\text{BE}}$[3:0]	I/O	PCI bus command/byte enable.	
	O	Outputs for the bi-directional command/byte enable.	
		State Meaning	Asserted/Negated—During the address phase, PCI_CBE[3:0],define the bus command. Byte enables determine which byte lanes carry meaningful data for PCI bus data phases. The PCI_CBE[0] signal applies to the LSB.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional command/byte enable.	
		State Meaning	Asserted/Negated—During the address phase, PCI_CBE[3:0], indicate the command that another master is sending. During the PCI bus data phase, PCI_CBE[3:0], indicate which byte lanes are valid.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_DEVSEL	I/O	PCI device select.	
	O	Outputs for the bi-directional device select.	
		State Meaning	Asserted—The PCI controller has decoded the address and is the target of the current access. Negated—The PCI controller has decoded the address and is not the target of the current access.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional device select.	
		State Meaning	Asserted—Some PCI agents (other than this PCI controller) have decoded its address as the target of the current access. Negated—No PCI agent has been selected.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_FRAME	I/O	PCI cycle frame. Used by the current PCI master to indicate the beginning and duration of an access.	
	O	Outputs for the bi-directional frame.	
		State Meaning	Asserted—The PCI controller acting as a PCI master which is initiating a bus transaction. While PCI_FRAME is asserted, data transfers may continue. Negated—If PCI_IRDY is asserted, indicates that the PCI transaction is in the final data phase; if PCI_IRDY is negated, indicates that the PCI bus is idle.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional frame.	
		State Meaning	Asserted—Another PCI master is initiating a bus transaction. Negated—The transaction is in the final data phase or that the bus is idle.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_GNT0	I/O	PCI arbiter grants. Output signal on this PCI controller when the arbiter is enabled. Input signal when the arbiter is disabled. **Note:** PCI_GNT[0] is a point-to-point signal. Every master has its own bus grant signal.	
	O	Outputs for the bi-directional arbiter grants.	
		State Meaning	Asserted—The PCI controller granted control of the PCI bus to agent 0. Negated—The PCI controller did not grant control of the PCI bus to agent 0.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional arbiter grants.	
		State Meaning	Asserted—The PCI controller has been granted control of the PCI bus by an external arbiter. Negated—The PCI controller has not been granted control of the PCI bus by an external arbiter.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_GNT[1:2]	O	PCI arbiter grants. Output signals on this PCI controller when the arbiter is enabled. Note that PCI_GNT*n* is a point-to-point signal. Every master has its own bus grant signal.	
		State Meaning	Asserted—The PCI controller granted control of the PCI bus to agent *n*. Negated—The PCI controller did not grant control of the PCI bus to agent *n*.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_IDSEL	I	PCI initialization device select. Used as a chip select during a PCI configuration cycle in agent mode. This signal should be tied low in host mode.	
		State Meaning	Asserted—The PCI controller is being selected as a target of a configuration read or write transactions. Negated—The PCI controller is not being selected as a target of configuration read or write transactions.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_INTA	O	PCI interrupt A.	
		State Meaning	Asserted—The PCI controller signals an interrupt to the PCI host. Negated—The PCI controller is not currently signalling an interrupt.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_IRDY	I/O	PCI initiator ready. This signal is driven by the PCI controller when it is the initiator of a PCI transfer.	
	O	Outputs for the bi-directional initiator ready.	
		State Meaning	Asserted—The PCI controller, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, this PCI controller asserts PCI_IRDY to indicate that valid data is present on PCI_AD[31:0]. During a read, this PCI controller asserts PCI_IRDY to indicate that it is prepared to accept data. Negated—The PCI target needs to wait before this PCI controller, acting as a PCI master, can complete the current data phase. During a write, this PCI controller negates PCI_IRDY to insert a wait cycle when it cannot provide valid data to the target. During a read, this PCI controller negates PCI_IRDY to insert a wait cycle when it cannot accept data from the target.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional initiator ready.	
		State Meaning	Asserted—Another PCI master can complete the current data phase of a transaction. Negated—If PCI_FRAME is asserted, indicates a wait cycle from another master. If PCI_FRAME is negated, indicates that the PCI bus is idle.

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_PAR	I/O	PCI parity.	
	O	Outputs for the bi-directional parity.	
		State Meaning	Asserted—Odd parity across PCI_AD[31:0] and PCI_CBE[3:0] during address and data phases. Negated—Even parity across PCI_AD[31:0] and PCI_AD[31:0] during address and data phases.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional parity.	
		State Meaning	Asserted—Odd parity driven by another PCI master or the PCI target during address and data phases. Negated—Even parity driven by another PCI master or the PCI target during address and data phases.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
$\overline{\text{PCI\_PERR}}$	I/O	PCI parity error	
	O	Outputs for the bi-directional parity error.	
		State Meaning	Asserted—The PCI controller, acting as a PCI agent, detected a data parity error. (driven by the PCI initiator on reads; driven by the PCI target on writes.) Negated—No error.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional parity error.	
		State Meaning	Asserted—Another PCI agent detects a data parity error while this PCI controller is sourcing data (this PCI controller was acting as the PCI initiator during a write, or is acting as the PCI target during a read). Negated—No error.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
$\overline{\text{PCI_REQ}}$0	I/O	PCI bus request**.** Input signal on this PCI controller when the arbiter is enabled. Output signal when the arbiter is disabled. Note that $\overline{\text{PCI\_REQ}}$*n* is a point-to-point signal. Every master has its own bus request signal.	
	O	Outputs for the bi-directional bus request.	
		State Meaning	Asserted—The PCI controller is requesting control of the PCI bus to perform a transaction. Negated—The PCI controller does not require use of the PCI bus.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Input for the bi-directional bus request.	
		State Meaning	Asserted—Agent 0 is requesting control of the PCI bus to perform a transaction. Negated—Agent 0 does not require use of the PCI bus.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_REQ[1:2]	I	PCI bus request. Input signals on this PCI controller when the arbiter is enabled. Note that PCI_REQ[n] is a point-to-point signal. Every master has its own bus request signal. Following is the state meaning for the PCI_REQ[n] input.	
		State Meaning	Asserted—An agent *n* is requesting control of the PCI bus to perform a transaction. Negated—An agent *n* does not require use of the PCI bus.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_RESET_OUT	O	PCI reset. This signal is used only in host mode. It should be left unconnected in agent mode.	
		State Meaning	Asserted—Devices on the PCI bus are in reset. Negated—Devices on the PCI bus operate normally.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_SERR	I/O	PCI system error	
	O	Outputs for the bi-directional system error.	
		State Meaning	Asserted—An address parity error, a target-abort (when this PCI controller is acting as the initiator), or some other system error (where the result is a catastrophic error) was detected. Negated—No error.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional system error.	
		State Meaning	Asserted—A device (other than this PCI controller) has detected a catastrophic error. Negated—No error.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
PCI_STOP	I/O	PCI stop.	
	O	Outputs for the bi-directional stop.	
		State Meaning	Asserted—The PCI controller, acting as a PCI target, is requesting that the initiator stop the current transaction. Negated—The current transaction can continue.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional stop.	
		State Meaning	Asserted—A target is requesting that this PCI controller, as the initiator, stop the current transaction. Negated—The current transaction can continue.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{PCI\_TRDY}}$	I/O	PCI target ready.	
	O	Outputs for the bi-directional target ready.	
		State Meaning	Asserted—The PCI controller, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, this PCI controller asserts $\overline{\text{PCI\_TRDY}}$ to indicate that valid data is present on PCI_AD[31:0]. During a write, this PCI controller asserts $\overline{\text{PCI\_TRDY}}$ to indicate that it is prepared to accept data. Negated—The PCI initiator needs to wait before this PCI controller, acting as a PCI target, can complete the current data phase. During a read, this PCI controller negates $\overline{\text{PCI\_TRDY}}$ to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates $\overline{\text{PCI\_TRDY}}$ to insert a wait cycle when it cannot accept data from the initiator.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3
	I	Inputs for the bi-directional target ready.	
		State Meaning	Asserted—Another PCI target is able to complete the current data phase of a transaction. Negated—A wait cycle from another target.
		Timing	Assertion/Negation—As specified by *PCI Local Bus Specification Rev* 2.3

13.3 Memory Map/Register Definitions

The PCI controller has the following types of registers:

- The PCI configuration access registers. Used for generating PCI configuration accesses from the CSB. These registers, listed in Table 13-4, are memory-mapped on the CSB and accessed through the IMMR window.
- The PCI memory-mapped registers. Used to manage error functions, general control and status, and address translation control for the inbound path. These registers are shown in Table 13-5. They can be accessed by PCI masters via the PCI controller to the CSB through the PIMMR inbound window. Note that Table 13-5 does not list outbound address translation registers; these are contained in the I/O sequencer (IOS) memory-mapped registers. SeeChapter 12, “DMA/Messaging Unit,” for more information.
- The PCI configuration space registers. Defined by the PCI specification. These registers are accessed by PCI masters using configuration accesses and are described in Section 13.3.3, “PCI Configuration Space Registers.”

Table 13-4. PCI Configuration Access Registers

Offset	Register	Access	Reset	Section/Page
PCI Configuration Access Registers				
0x0	PCI_CONFIG_ADDRESS	W	0x0000_0000	13.3.1.1/13-13
0x4	PCI_CONFIG_DATA	R/W	0x0000_0000	13.3.1.2/13-14
0x8	PCI_INT_ACK	R	—	13.3.1.3/13-15

Table 13-5. PCI Memory-Mapped Registers

Offset	Register	Access	Reset	Section/Page
PCI Error Management Registers				
0x00	PCI error status register (PCI_ESR)	w1c	0x0000_0000	13.3.2.1/13-15
0x04	PCI error capture disable register (PCI_ECDR)	R/W	0x0000_0000	13.3.2.2/13-16
0x08	PCI error enable register (PCI_EER)	R/W	0x0000_0000	13.3.2.3/13-17
0x0C	PCI error attributes capture register (PCI_EATCR)	R/W	0x0000_0000	13.3.2.4/13-18
0x10	PCI error address capture register (PCI_EACR)	R	0x0000_0000	13.3.2.5/13-19
0x14	PCI error extended address capture register (PCI_EEACR)	R	0x0000_0000	13.3.2.6/13-20
0x18	PCI error data capture register (PCI_EDCR)	R/W	0x0000_0000	13.3.2.7/13-20
PCI Control and Status Registers				
0x20	PCI general control register (PCI_GCR)	R/W	0x0000_0000	13.3.2.8/13-21
0x24	PCI error control register (PCI_ECR)	R/W	0x0000_0000	13.3.2.9/13-21
0x28	PCI general status register (PCI_GSR)	R	0x0000_0000	13.3.2.10/13-22
PCI Inbound ATU Registers				
0x38	PCI inbound translation address register 2 (PITAR2)	R/W	0x0000_0000	13.3.2.11/13-23
0x3C	Reserved	—	—	—
0x40	PCI inbound base address register 2 (PIBAR2)	R/W	0x0000_0000	13.3.2.12/13-24
0x44	PCI inbound extended base address register 2 (PIEBAR2)	R/W	0x0000_0000	13.3.2.13/13-24
0x48	PCI inbound window attributes register 2 (PIWAR2)	R/W	0x0000_0000	13.3.2.14/13-25
0x50	PCI inbound translation address register 1 (PITAR1)	R/W	0x0000_0000	13.3.2.11/13-23
0x54	Reserved	—	—	—
0x58	PCI inbound base address register 1 (PIBAR1)	R/W	0x0000_0000	13.3.2.12/13-24
0x5C	PCI inbound extended base address register 1 (PIEBAR1)	R/W	0x0000_0000	13.3.2.13/13-24
0x60	PCI inbound window attributes register 1 (PIWAR1)	R/W	0x0000_0000	13.3.2.14/13-25
0x68	PCI inbound translation address register 0 (PITAR0)	R/W	0x0000_0000	13.3.2.11/13-23
0x6C	Reserved	—	—	—
0x70	PCI inbound base address register 0 (PIBAR0)	R/W	0x0000_0000	13.3.2.12/13-24
0x78	PCI inbound window attributes register 0 (PIWAR0)	R/W	0x0000_0000	13.3.2.13/13-24
0x7C–0xFF	Reserved	—	—	—

13.3.1 PCI Configuration Access Registers

This section describes the registers used to allow a local bus master to access the PCI configuration space, and generate special cycle or interrupt acknowledge transactions on the PCI bus. A special case provides

access to the PCI controller's internal PCI configuration registers. The PCI registers, PCI_CONFIG_ADDRESS, PCI_CONFIG_DATA, and PCI_INT_ACK, are little endian registers.

13.3.1.1 PCI_CONFIG_ADDRESS

Figure 13-3 shows the PCI_CONFIG_ADDRESS register fields.

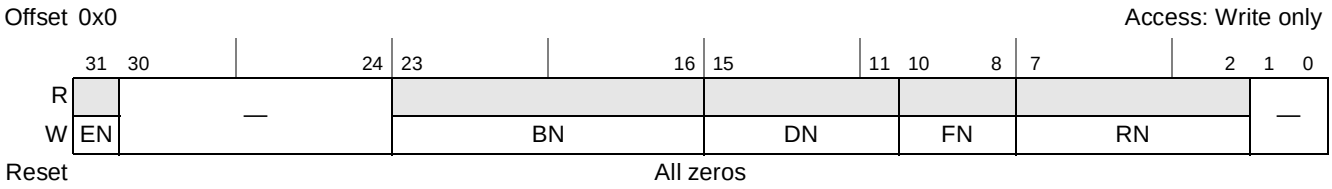


Figure 13-3. PCI_CONFIG_ADDRESS Register

The PCI_CONFIG_ADDRESS register holds the address for an access to the PCI configuration space from the local bus. This register must be programmed before accessing PCI_CONFIG_DATA to perform the transaction. Only 32-bit accesses are permitted.

If EN=1, BN=0, and DN=0, the access is to the internal PCI configuration registers, so no transaction is generated on the PCI bus.

If EN=1, BN=0, DN=31, FN=7, and RN=0, writing to PCI_CONFIG_DATA generates a special cycle transaction and reading from PCI_CONFIG_DATA generates an interrupt acknowledge transaction.

Table 13-6 shows the bit settings of the PCI_CONFIG_ADDRESS register.

Table 13-6. PCI_CONFIG_ADDRESS Field Descriptions

Bits	Name	Description
31	EN	Enable configuration transaction. Determines the type of transaction to be generated. 0 No configuration transaction will be generated by accessing the CONFIG_DATA register. Such an access will be passed through to the PCI bus as an I/O transaction. Because this is generally not desirable, the user should not access CONFIG_DATA when the EN bit is 0. 1 A configuration transaction will be generated by accessing the CONFIG_DATA register if BN and DN are not both zero.
30–24	—	Reserved
23–16	BN	Bus number. Specifies the bus segment to which a configuration transaction is directed. If this field is 0, a Type 0 configuration transaction is generated. Otherwise, a Type 1 configuration transaction is generated.

Table 13-6. PCI_CONFIG_ADDRESS Field Descriptions (continued)

Bits	Name	Description
15–11	DN	Device number. Specifies the device to which a configuration transaction is directed. For a Type 0 configuration transaction, this field is decoded to individual PCI1_IDSEL signals for the address phase according to the following values. For a Type 1 configuration transaction, this field is used directly for the address phase. **Value** — **AD Signal that is Driving High** 01010 — 31 01011 — 11 01100 — 12 01101 — 13 01110 — 14 01111 — 15 10000 — 16 10001 — 17 10010 — 18 10011 — 19 10100 — 20 10101 — 21 10110 — 22 10111 — 23 11000 — 24 11001 — 25 11010 — 26 11011 — 27 11100 — 28 11101 — 29 11110 — 30 11111 — Special cycle / interrupt acknowledge 00000 — Internal access Others — Reserved
10–8	FN	Function number. Specifies the function to which the configuration transaction is directed on a multi-function device. It is used directly in the address phase of the configuration transaction.
7–2	RN	Register number. Specifies the register being accessed in the PCI configuration space.
1–0	—	Reserved

13.3.1.2 PCI_CONFIG_DATA

An access to PCI_CONFIG_DATA usually generates a PCI configuration transaction if PCI_CONFIG_ADDRESS[EN] is set. There are some exceptions contained in the description of PCI_CONFIG_ADDRESS[EN].

This register may be accessed with an 8-, 16-, or 32-bit access, depending on the width of the register targeted by the configuration transaction.

Figure 13-4 shows the PCI_CONFIG_DATA register fields.

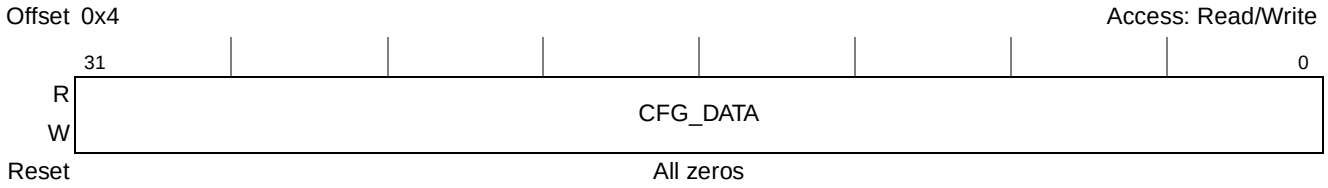


Figure 13-4. PCI_CONFIG_DATA

Table 13-7 shows the bit settings of the PCI_CONFIG_DATA register.

Table 13-7. PCI_CONFIG_DATA Field Descriptions

Bits	Name	Description
31–0	CFG_DATA	Configuration data. This field contains the data transferred on a PCI configuration transaction.

13.3.1.3 PCI Interrupt Acknowledge Register (PCI_INT_ACK)

Reading this register generates an interrupt acknowledge transaction on the PCI bus. The value that is read is undefined.

13.3.2 PCI Memory-Mapped Control and Status Registers

This section describes the control and status registers.

13.3.2.1 PCI Error Status Register (PCI_ESR)

The PCI error status register (PCI_ESR) contains status bits for various types of error conditions captured by the PCI controller. Each status bit is set when the corresponding error condition is captured. PCI_ESR is a write-1-to-clear type register. A bit is cleared whenever the register is written and the data in the corresponding bit location is a 1. Figure 13-5 shows the PCI_ESR fields.

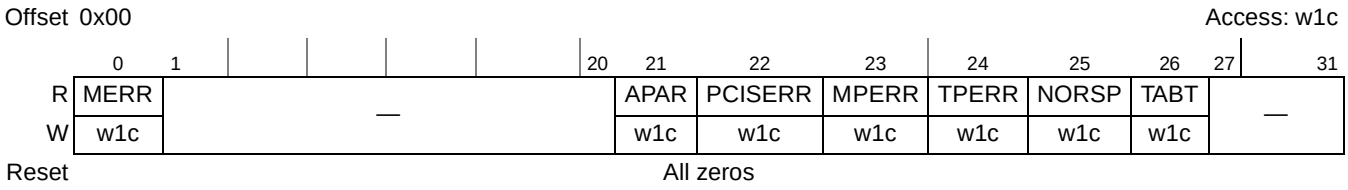


Figure 13-5. PCI Error Status Register (PCI_ESR)

Table 13-8 describes the bit settings of the PCI_ESR register.

Table 13-8. PCI_ESR Field Descriptions

Bits	Name	Description
0	MERR	Multiple errors. Set if any other bit of this register is 1 and the same error type occurs again.
1–20	—	Reserved
21	APAR	Address parity error. Set when there is an address parity error on a PCI access initiated by a device other than this PCI controller.
22	PCISERR	PCI system error.Set when the $\overline{\text{PCI\_SERR}}$ input signal is asserted. See Table 13-3 for more information on $\overline{\text{PCI\_SERR}}$.
23	MPERR	Master parity error. Set when the $\overline{\text{PCI\_PERR}}$ input signal is asserted on a write access initiated by this PCI controller or when a data parity error is detected by this PCI controller on a read access that it initiated.
24	TPERR	Target parity error. Set when this PCI controller is the target of a transaction and the $\overline{\text{PCI\_PERR}}$ input signal is asserted on a read access or a data parity error is detected by this PCI controller on a write access.
25	NORSP	No response. Set when there is no response to a transaction initiated by this PCI controller on the PCI bus (no $\overline{\text{PCI\_DEVSEL}}$ assertion).
26	TABT	Target abort. Set when a PCI target abort occurs on a transaction initiated by this PCI controller.
27–31	—	Reserved

13.3.2.2 PCI Error Capture Disable Register (PCI_ECDR)

PCI_ECDR contains fields for controlling the capture of the transaction that caused an error. Each bit corresponds to the error condition reported in the PCI error status register (PCI_ESR). Note that only the first error is captured, so disabling the capture of some error types may allow greater visibility of the significant errors.

- 1 = Do not capture the transaction that caused this error.
- 0 = Capture the transaction that caused this error.

Figure 13-6 shows the PCI_ECDR fields.

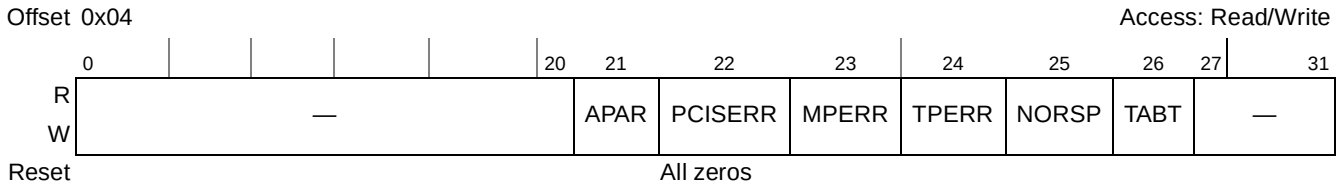


Figure 13-6. PCI Error Capture Disable Register (PCI_ECDR)

Table 13-9 describes the bit settings of the PCI_ECDR register.

Table 13-9. PCI_ECDR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	APAR	Address parity error. Disable capture for address parity errors
22	PCISERR	PCI system error. Disable capture for received $\overline{\text{PCI\_SERR}}$ errors
23	MPERR	Master parity error. Disable capture for master $\overline{\text{PCI\_PERR}}$ errors
24	TPERR	Target parity error. Disable capture for target $\overline{\text{PCI\_PERR}}$ errors
25	NORSP	No response. Disable capture for master-abort errors
26	TABT	Target abort. Disable capture for target abort errors
27–31	—	Reserved

13.3.2.3 PCI Error Enable Register (PCI_EER)

PCI_EER contains fields for enabling the assertion of an interrupt for the error conditions reported in the PCI error status register (PCI_ESR).

- 1 = The interrupt is enabled.
- 0 = The interrupt is disabled.

Figure 13-7 shows the PCI_EER fields.

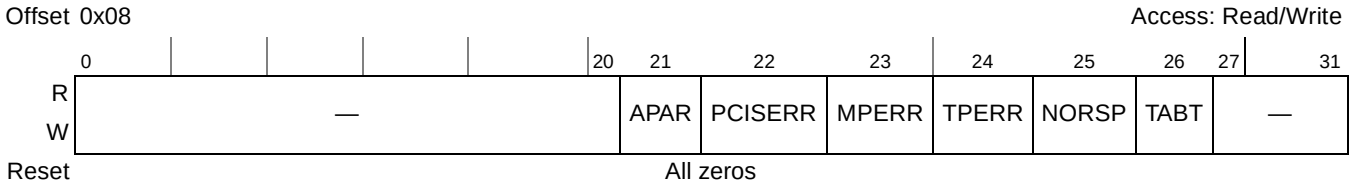


Figure 13-7. PCI Error Enable Register (PCI_EER)

Table 13-10 describes the bit settings of the PCI_EER register.

Table 13-10. PCI_EER Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	APAR	Address parity error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
22	PCISERR	PCI system error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
23	MPERR	Master parity error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
24	TPERR	Target parity error. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
25	NORSP	No response. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.

Table 13-10. PCI_EER Field Descriptions (continued)

Bits	Name	Description
26	TABT	Target abort. Generate an interrupt when the corresponding bit of the PCI_ESR is 1.
27–31	—	Reserved

13.3.2.4 PCI Error Attributes Capture Register (PCI_EATCR)

PCI_EATCR contains fields for storing information associated with the first PCI error captured. Figure 13-8 shows the PCI_EATCR fields.

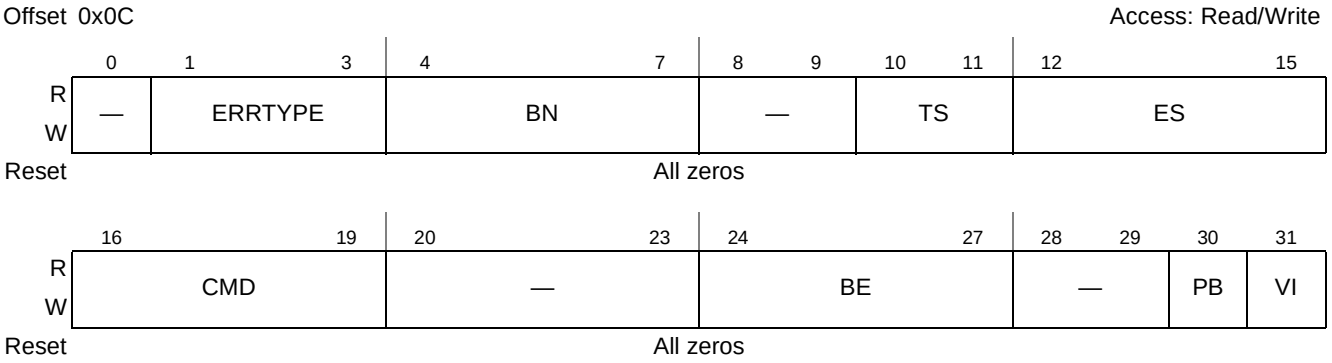


Figure 13-8. PCI Error Attributes Capture Register (PCI_EATCR)

Table 13-11 describes the bit settings of the PCI_EATCR register.

Table 13-11. PCI_EATCR Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	ERRTYPE	First error type. This field is encoded to indicate the type of the first PCI error captured. 000 Address parity error 001 Write data parity error 010 Read data parity error 011 Master abort 100 Target abort 101 System error indication received 110 Parity error indication received on a read 111 Parity error indication received on a write
4–7	BN	Beat number. This field provides the data beat number on which the error occurred for data parity errors. The value of this field is undefined for other error types. The beat values are described as follows: 0000 1st beat 0001 2nd beat 0010 3rd beat 0011 4th beat 0100 5th beat 0101 6th beat 0110 7th beat 0111 8th beat 1000 9th beat or beyond (transaction larger than one cache line) Others Reserved

Table 13-11. PCI_EATCR Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved
10–11	TS	Transaction size. Indicates the size of the transaction in units of doublewords (8 bytes). If the transaction crossed a cache line (32-byte) boundary, this field indicates the number of actual double words in the cache line on which the error occurred. This field is valid only if the PCI controller was the master of the transaction. 00 4 double words 01 1 double word 10 2 double words 11 3 double words
12–15	ES	Error source. This field indicates the source of the PCI transaction. 0000 External master 0101 DMA Others reserved
16–19	CMD	PCI command. Contains the PCI command PCI_CBE[3:0] of the transaction.
20–23	—	Reserved
24–27	BE	PCI byte enables. Contains the PCI byte enables PCI_CBE[3:0] for the data word.
28–29	—	Reserved
30	PB	Parity bit. Contains the PCI parity bit for the captured data word.
31	VI	Error information valid. This bit indicates that the error information captured in this register, PCI_EACR, PCI_EEACR, and PCI_EDCR is valid. 0 No valid error information 1 Error information is valid

13.3.2.5 PCI Error Address Capture Register (PCI_EACR)

PCI_EACR contains fields for storing the low portion of the address associated with the first PCI error captured. Figure 13-9 shows the PCI_EACR fields.

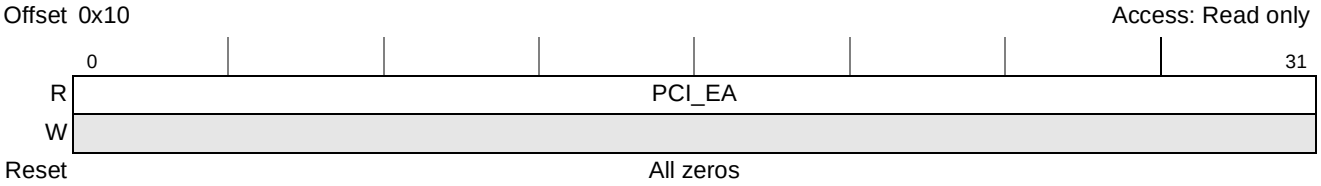


Figure 13-9. PCI Error Address Capture Register (PCI_EACR)

Table 13-12 describes the bit settings of the PCI_EACR register.

Table 13-12. PCI_EACR Field Description

Bits	Name	Description
0–31	PCI_EA	PCI error address. Contains the low portion of the address associated with the first detected error. Read only.

13.3.2.6 PCI Error Extended Address Capture Register (PCI_EEACR)

PCI_EEACR contains fields for storing the high portion of the address associated with the first PCI error captured. Figure 13-10 shows the PCI_EEACR fields.

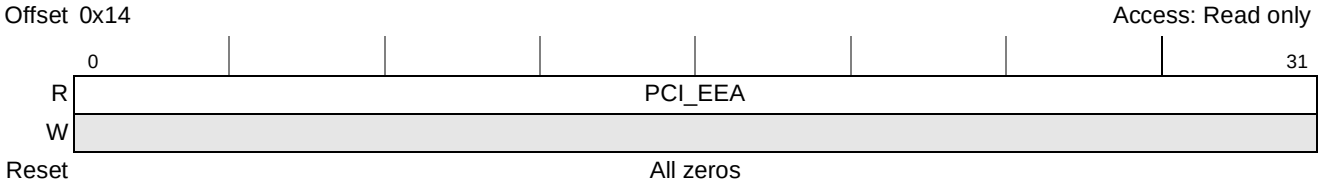


Figure 13-10. PCI Error Extended Address Capture Register (PCI_EEACR)

Table 13-13 describes the bit settings of the PCI_EEACR register.

Table 13-13. PCI_EEACR Field Description

Bits	Name	Description
0–31	PCI_EEA	PCI error extended address. Contains the high portion of the address associated with the first detected error.

13.3.2.7 PCI Error Data Low Capture Register (PCI_EDLCR)

PCI_EDLCR contains fields for storing the data associated with the first PCI error captured. Figure 13-11 shows the PCI_EDLCR fields.

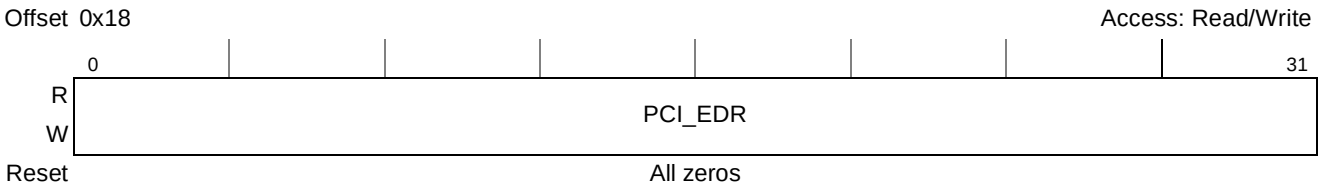


Figure 13-11. PCI Error Data Low Capture Register (PCI_EDLCR)

Table 13-14 describes the bit settings of the PCI_EDLCR register.

Table 13-14. PCI_EDLCR Field Description

Bits	Name	Description
0–31	PCI_EDR	PCI error data. Contains the data associated with the first detected error.

13.3.2.8 PCI General Control Register (PCI_GCR)

PCI_GCR contains fields for controlling the behavior of the internal arbiter, the state of the bus signals, and the PCI reset signal for host mode. Figure 13-12 shows the PCI_GCR fields.

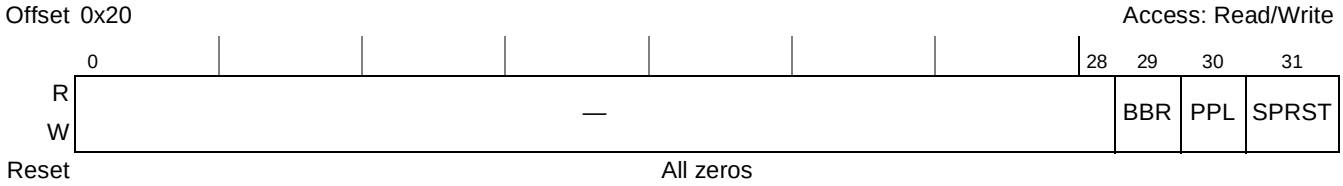


Figure 13-12. PCI General Control Register (PCI_GCR)

Table 13-15 shows the bit settings of PCI_GCR. The bits that are not reserved have read and write capability.

Table 13-15. PCI_GCR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	BBR	Block bus requests. This bit could be used to prepare for entering a low-power mode by preventing transactions on the PCI bus. 0 External bus requests are treated normally. 1 Block external bus requests. When this bit is set, all bus requests from external devices to the PCI controller's internal arbiter are blocked, and the bus is continuously granted to the PCI controller.
30	PPL	PCI pins low. This bit could be used to put the bus signals in a safe electrical state when the devices on the bus are powered down. This bit should never be set during normal operation of the PCI bus. 0 PCI pins function normally 1 PCI pins in the low state. Setting this bit forces all the output and bidirectional pins of the PCI bus to be driven low.
31	SPRST	Soft PCI reset. This bit provides software control of the $\overline{\text{PCI\_RESET\_OUT}}$ output signal. It is only valid in host mode. 0 $\overline{\text{PCI\_RESET\_OUT}}$ is driven low. 1 $\overline{\text{PCI\_RESET\_OUT}}$ is driven high.

13.3.2.9 PCI Error Control Register (PCI_ECR)

PCI_ECR contains fields for determining whether an interrupt or machine check is generated for the error conditions reported in the PCI error status register (PCI_ESR). Note that if the corresponding bit in the PCI error enable register (PCI_EER) is clear, the bit in the PCI error control register (PCI_ECR) has no effect.

1 = A machine check is generated.
0 = An interrupt is generated.

Figure 13-13 shows the PCI_ECR fields.

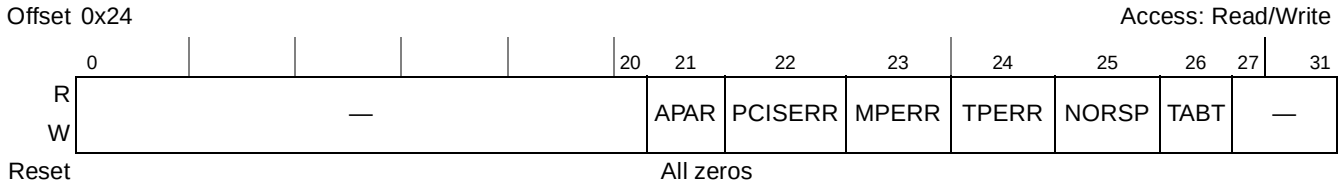


Figure 13-13. PCI Error Control Register (PCI_ECR)

Table 13-16 describes the bit settings of the PCI_ECR register.

Table 13-16. PCI_ECR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	APAR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
22	PCISERR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
23	MPERR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
24	TPERR	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
25	NORSP	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
26	TABT	0 An interrupt is generated if the corresponding bit of the PCI_ESR is 1. 1 A machine check is generated if the corresponding bit of the PCI_ESR is 1.
27–31	—	Reserved

13.3.2.10 PCI General Status Register (PCI_GSR)

PCI_GSR contains fields for providing status information, shown in Figure 13-14.

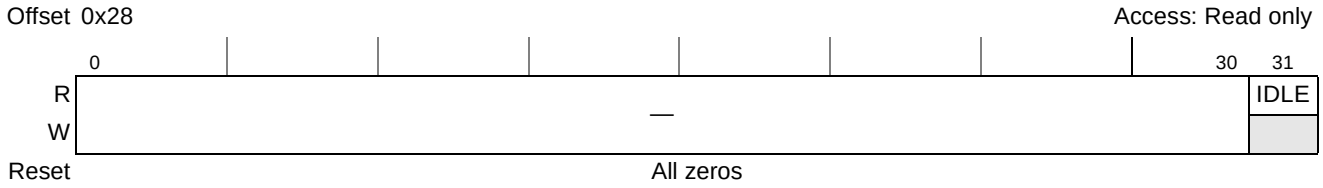


Figure 13-14. PCI General Status Register (PCI_GSR)

Table 13-17 shows the bit settings of the PCI_GSR register. All bits are read-only.

Table 13-17. PCI_GSR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	IDLE	PCI controller is idle. Indicates when the PCI bus is totally idle before setting PCI_GCR[PPL]. 0 The PCI controller is active. 1 The PCI controller is idle.

13.3.2.11 PCI Inbound Translation Address Registers (PITAR*n*)

PITAR*n* contains fields for defining the starting point of the inbound translation windows in the local memory space (see Section 13.4.6, “PCI Inbound Address Translation”; for more information on outbound address translation registers, see Chapter 11, “Sequencer”). Inbound and outbound windows for the same bus should not overlap. Therefore, situations where an inbound window translation points back into an outbound window, or where an outbound translation window points back into an inbound window, are not allowed.

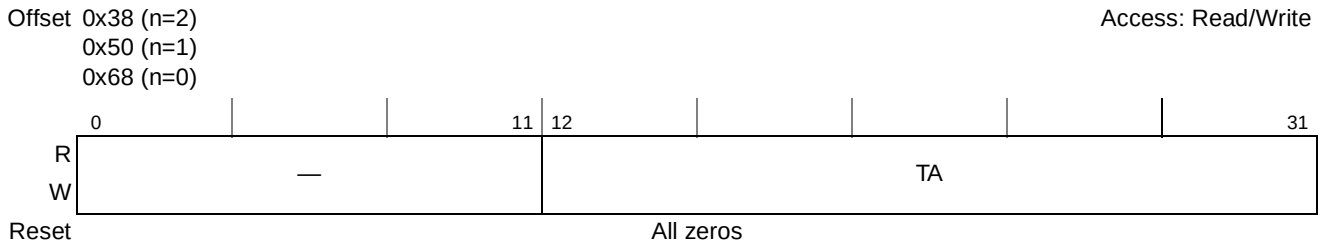


Figure 13-15. PCI Inbound Translation Address Registers (PITAR*n*)

Table 13-18 shows the bit settings of PITAR*n*.

Table 13-18. PITAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TA	Translation address. Contains the starting address of the inbound translated address. TA corresponds to the 20 highest-order bits of a 32-bit local address. The specified address must be aligned to the window size, as defined by PIWAR*n*[IWS].

13.3.2.12 PCI Inbound Base Address Registers (PIBAR*n*)

PIBAR*n* contains fields for defining the starting point of the inbound windows in the PCI memory space. A write to a PIBAR*n* register also causes a change in the base address bits in the corresponding GPL base address register in the PCI configuration space. Figure 13-16 shows the PIBARx fields.

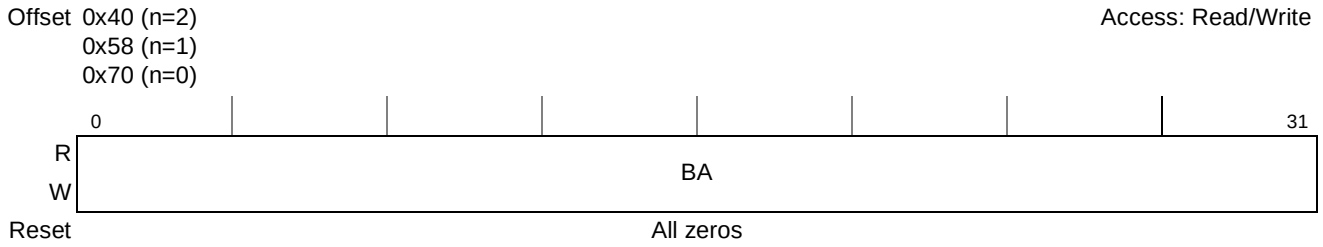


Figure 13-16. PCI Inbound Base Address Registers (PIBAR*n*)

Table 13-19 shows the bit settings of PIBAR*n*.

Table 13-19. PIBAR*n* Field Descriptions

Bits	Name	Description
0–31	BA	Base address. Contains the starting address in the PCI memory space of the inbound window. This field corresponds to bits 43–12 of a 64-bit address. In PIBAR0, the upper 12 bits are reserved because only a 32-bit address is supported. The specified address must be aligned to the window size, as defined by PIWAR*n*[IWS].

13.3.2.13 PCI Inbound Extended Base Address Registers (PIEBAR*n*)

PIEBAR*n* contains fields for defining the high portion of the starting point of the inbound windows in the PCI memory space. Figure 13-17 shows the PIEBAR*n* fields.

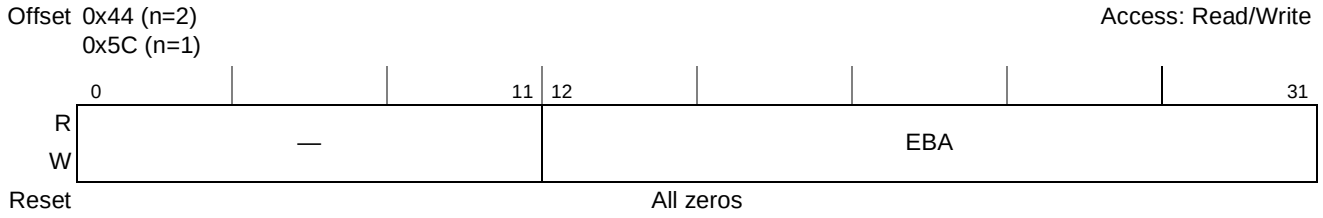


Figure 13-17. PCI Inbound Extended Base Address Registers (PIEBAR*n*)

Table 13-20 shows the bit settings of PIEBAR*n*.

Table 13-20. PIEBAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	EBA	Extended base address. Contains the high portion of the starting address in the PCI memory space of the inbound base address. This 20-bit field corresponds to bits 63–44 of a 64-bit address.

13.3.2.14 PCI Inbound Window Attribute Registers (PIWAR*n*)

PIWAR*n* contains fields for defining the size of an inbound translation window. It also defines some properties of the window. Figure 13-18 shows the PIWAR*n* fields. SeeSection 4.3.1.1, "Reset Configuration Word Source," for more information on reset configuration.

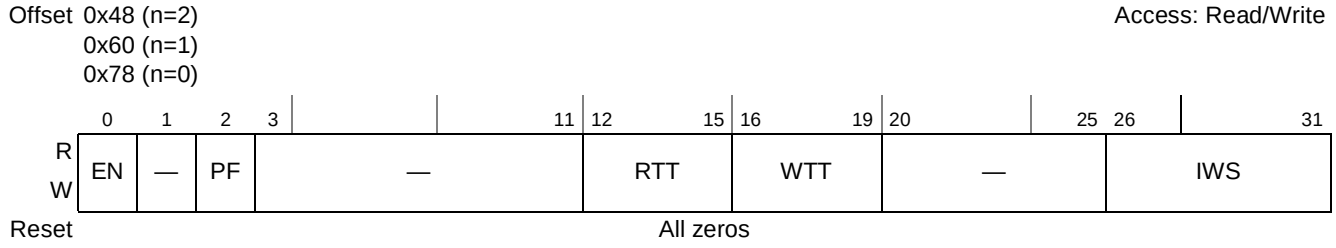


Figure 13-18. PCI Inbound Window Attribute Registers (PIWAR*n*)

Table 13-21 shows the bit settings of PIWAR*n*.

Table 13-21. PIWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. Used to enable the address translation window. 0 Address translation is disabled for this window. 1 Address translation is enabled for this window. PCI addresses that match the definition of the window will be recognized by the PCI controller and translated to the local memory space.
1	—	Reserved
2	PF	Prefetchable. Defines whether the transactions that are translated through this window are prefetchable on the local bus. Streaming the transactions requires the memory space to be prefetchable. 0 Not prefetchable 1 Prefetchable
3–11	—	Reserved
12–15	RTT	Read transaction type. Determines the type of transaction performed on the local bus when the PCI transaction is a read. The RTT values are described as follows: 0100 Read without snoop on system bus 0101 Read with snoop on system bus Others reserved
16–19	WTT	Write transaction type. Determines the type of transaction performed on the local bus when the PCI transaction is a write. The WTT values are described as follows: 0100 Write without snoop of local processor 0101 Write with snoop of local processor Others reserved

Table 13-21. PIWAR*n* Field Descriptions (continued)

Bits	Name	Description
20–25	—	Reserved
26–31	IWS	Inbound window size. Indicates the size of the inbound translation window. Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window is 4 Kbytes (N = 11) 000000–001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011110 2-Gbyte window size 011111–111111 Reserved

13.3.3 PCI Configuration Space Registers

This section describes the PCI configuration space registers. These registers are shown with descending bit numbering to correspond to the PCI standard.

NOTE

The registers described in this section use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data. No byte swapping occurs when the registers are accessed from the PCI bus.

Table 13-22 shows the PCI configuration registers that are mapped in PCI configuration space. Some fields are common to registers in both spaces to ensure consistency. These fields are discussed in the register definitions.

Table 13-22. PCI Configuration Space Registers

Address	Use	Access
00	Vendor ID configuration register	R
02	Device ID configuration register	R
04	PCI command configuration register	R/W
06	PCI status configuration register	Read/bit-reset
08	Revision ID configuration register	R
09	Standard programming interface	R
0A	Subclass code configuration register	R
0B	Base class code configuration register	R
0C	Cache line size configuration register	R/W
0D	Latency timer configuration register	R/W
0E	Header type configuration register	R
0F	BIST control configuration register	R

Table 13-22. PCI Configuration Space Registers (continued)

Address	Use	Access
10	PIMMR base address register	R/W
14	GPL base address register 0	R/W
18	GPL base address register 1	R/W
1C	GPL extended base address register 1	R/W
20	GPL base address register 2	R/W
24	GPL extended base address register 2	R/W
2C	Subsystem vendor ID configuration register	R
2E	Subsystem device ID configuration register	R
34	Capabilities pointer configuration register	R
3C	Interrupt line configuration register	R/W
3D	Interrupt pin configuration register	R
3E	Minimum grant configuration register	R
3F	Maximum latency configuration register	R
44	PCI function configuration register	R/W
46	PCI arbiter control register (PCIACR)	R/W
48	Hot swap register block	R/W

13.3.3.1 Vendor ID Configuration Register

Figure 13-19 shows the vendor ID fields. This is a read-only register.

Offset 0x00 Access: Read only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VID															
W																
Reset	0	0	0	1	1	0	0	1	0	1	0	1	0	1	1	1

Figure 13-19. Vendor ID Configuration Register

Table 13-23 shows the bit settings of the vendor ID register.

Table 13-23. Vendor ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	VID	Vendor ID. The read-only value 0x1957 specifies Freescale Semiconductor as the manufacturer of the device.

13.3.3.2 Device ID Configuration Register

Figure 13-20 shows the device ID fields. This is a read only register.

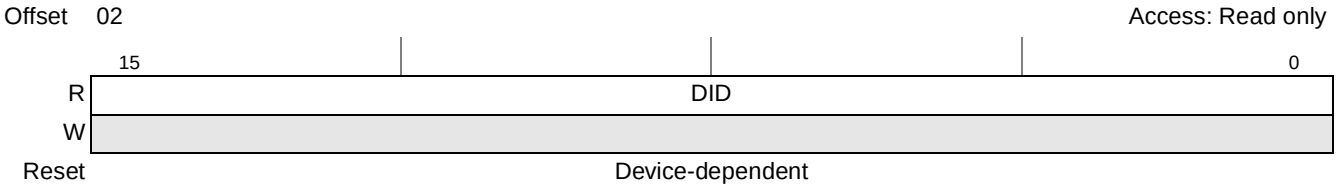


Figure 13-20. Device ID Configuration Register

Table 13-24 shows the bit settings of the device ID register.

Table 13-24. Device ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	DID	Device ID. This field identifies the device. 00A0 Reserved 00A1 Reserved 00A2 MPC8323E 00A3 MPC8323 00A6 MPC8321E 00A7 MPC8321

13.3.3.3 PCI Command Configuration Register

Figure 13-21 shows the PCI command fields.

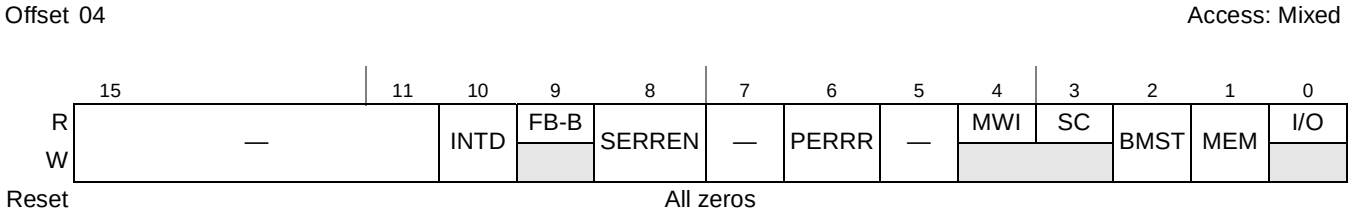


Figure 13-21. PCI Command Configuration Register

Table 13-25 shows the bit settings of the PCI command register.

Table 13-25. PCI Command Configuration Register Field Descriptions

Bits	Name	Description
15–11	—	Reserved
10	INTD	Interrupt Disable. Setting this bit masks the $\overline{\text{PCI\_INTA}}$ output. 0 $\overline{\text{PCI\_INTA}}$ provides the device interrupt status. 1 $\overline{\text{PCI\_INTA}}$ is always negated.
9	FB-B	Fast back-to-back. Hard-wired to 0.

Table 13-25. PCI Command Configuration Register Field Descriptions (continued)

Bits	Name	Description
8	SERREN	SERR enable. This bit is an enable bit for the SERR driver. Address parity errors are reported only if this bit and bit 6 are 1. 0 $\overline{\text{PCI\_SERR}}$ is never asserted. 1 $\overline{\text{PCI\_SERR}}$ may be asserted to indicate error conditions.
7	—	Reserved
6	PERRR	Parity error response. Controls the PCI controller's response to a parity error. 0 Parity errors are ignored and normal operation continues. 1 Standard parity error treatment.
5	—	Reserved
4	MWI	Memory-write-and-invalidate. Hard-wired to 0.
3	SC	Special cycles. Hard-wired to 0.
2	BMST	Bus master. Controls the PCI controller's ability to be a master on the PCI bus. At reset, this bit is cleared in Agent Mode and set in Host Mode. 0 The PCI controller does not generate PCI accesses. 1 The PCI controller behaves as a bus master.
1	MEM	Memory space. Controls the response to memory space accesses. 0 The PCI controller does not respond to Memory Space accesses. 1 The PCI controller as a target responds to Memory Space accesses.
0	I/O	I/O space. Hard-wired to 0.

13.3.3.4 PCI Status Configuration Register

This register is used to record status information for PCI bus-related events. Some of the bits are hard-wired to indicate the capabilities of the PCI controller. Other bits can be cleared by writing 1 to the bit location. Figure 13-22 shows the PCI status fields.

Offset 06 Access: Mixed

	15	14	13	12	11	10	9	8	7	6	5	4	3	2		0
R	DPERR	SSERR	RMA	RTA	STA	DEVSEL_T		DPD	FB-BC	—	66M	CL	INTS	—		
W	w1c	w1c	w1c	w1c	w1c			w1c					w1c			
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0

Figure 13-22. PCI Status Configuration Register

Table 13-26 shows the bit settings of the PCI status register.

Table 13-26. PCI Status Configuration Register Field Descriptions

Bits	Name	Description
15	DPERR	Detected parity error. Set whenever the PCI controller detects a parity error on the PCI bus, even if parity error handling is disabled (as controlled by bit 6 in the PCI Command register).
14	SSERR	Signaled system error. Set whenever $\overline{\text{PCI\_SERR}}$ is asserted.

Table 13-26. PCI Status Configuration Register Field Descriptions (continued)

Bits	Name	Description
13	RMA	Received master abort. Set whenever the PCI controller, acting as the PCI master on the PCI bus, terminates a transaction (except for a special-cycle) using master-abort.
12	RTA	Received target abort. Set whenever a transaction initiated by this PCI controller on the PCI bus is terminated by a target-abort.
11	STA	Signaled target abort. Set whenever the PCI controller, acting as the PCI target on the PCI bus, issues a target-abort to a PCI master.
10–9	DEVSEL_T	DEVSEL timing. Hard-wired to 00.
8	DPD	Master data parity error. Set when a data parity error is detected on the PCI bus, if the PCI controller is the master that initiated the transaction and bit 6 in the PCI command register is set.
7	FB-BC	Fast back-to-back capable. Hard-wired to 1.
6	—	Reserved
5	66M	66-MHz capable. Hard-wired to 1.
4	CL	Capabilities list. Hard-wired to 1.
3	INTS	Interrupt status. Contains the status of the device interrupt. The value of this bit is not affected by the INTD bit of the PCI command configuration register.
2–0	—	Reserved

13.3.3.5 Revision ID Configuration Register

Figure 13-23 shows the revision ID fields.

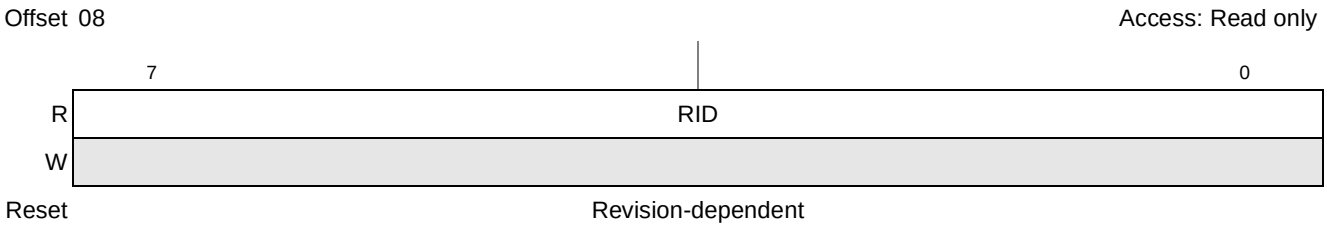


Figure 13-23. Revision ID Configuration Register

Table 13-27 shows the bit settings of the revision ID register.

Table 13-27. Revision ID Configuration Register Field Descriptions

Bits	Name	Description
7–0	RID	Revision ID. Specifies a revision code of the PCI controller.

13.3.3.6 Standard Programming Interface Configuration Register

Figure 13-24 shows the standard programming interface fields. This is the lower byte of the class code.

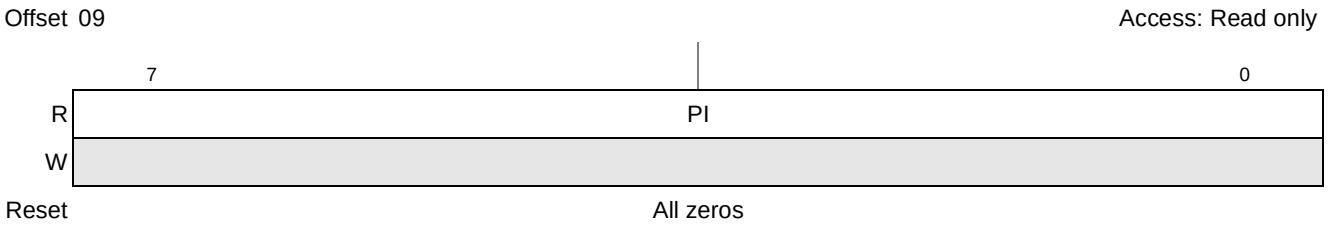


Figure 13-24. Standard Programming Interface Configuration Register

Table 13-28 shows the bit settings of the standard programming interface register.

Table 13-28. Standard Programming Interface Configuration Register Field Descriptions

Bits	Name	Description
7–0	PI	Programming interface. This field is hard-wired to 0x00.

13.3.3.7 Subclass Code Configuration Register

Figure 13-25 shows the subclass code fields. This is the middle byte of the class code.

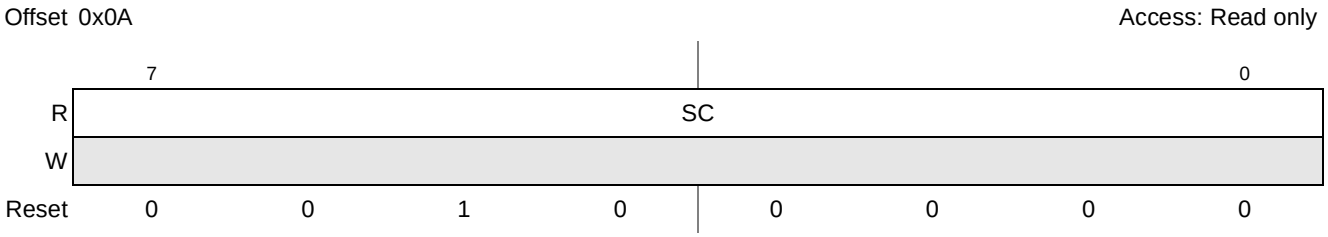


Figure 13-25. Subclass Code Configuration Register

Table 13-29 shows the bit settings of the subclass code register.

Table 13-29. Subclass Code Configuration Register Field Descriptions

Bits	Name	Description
7–0	SC	Sub-class code. This field is hard-wired to 0x20, indicating a Power PC processor.

13.3.3.8 Base Class Code Configuration Register

Figure 13-26 shows the base class code fields. This is the upper byte of the class code.

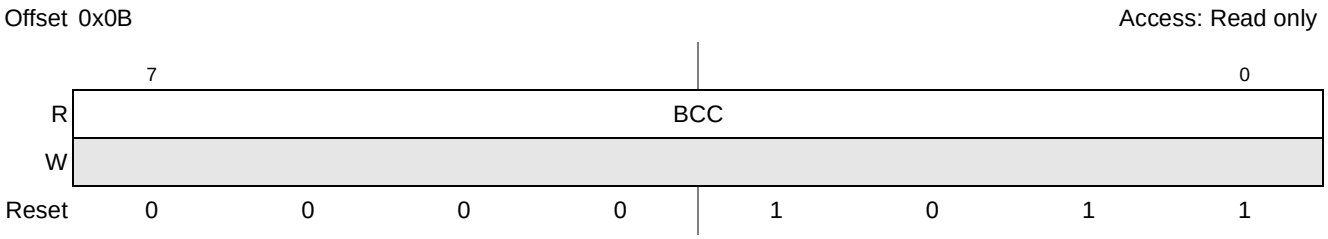


Figure 13-26. Base Class Code Configuration Register

Table 13-30 shows the bit settings of the class code register.

Table 13-30. Class Code Configuration Register Field Descriptions

Bits	Name	Description
7–0	BCC	Base class code. This field is hard-wired to 0x0B, indicating a processor.

13.3.3.9 Cache Line Size Configuration Register

Figure 13-27 shows the cache line size fields.

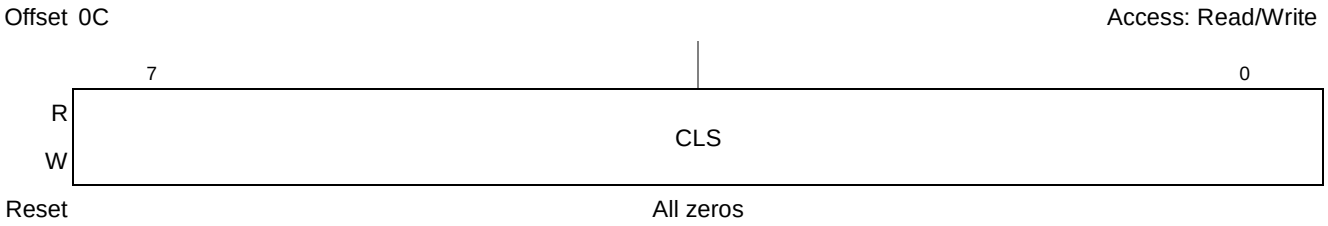


Figure 13-27. Cache Line Size Configuration Register

Table 13-31 shows the bit settings of the cache line size register.

Table 13-31. Cache Line Size Configuration Register Field Descriptions

Bits	Name	Description
7–0	CLS	Cache line size. Cache-line in terms of 32-bit words. Although the register is writable, only the value 0x08 is legal.

13.3.3.10 Latency Timer Configuration Register

Figure 13-28 shows the latency timer fields.

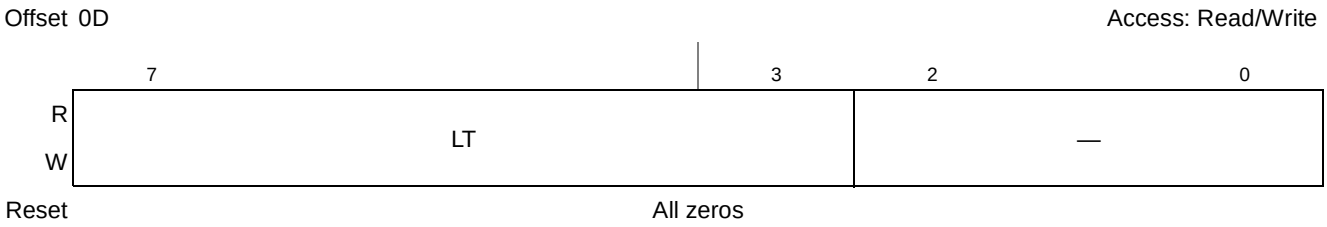


Figure 13-28. Latency Timer Configuration Register

Table 13-32 shows the bit settings of the latency timer register.

Table 13-32. Latency Timer Configuration Register Field Descriptions

Bits	Name	Description
7–3	LT	Latency timer. Specifies a granularity of 8 PCI clocks, the length of time that the PCI controller, when mastering a transaction, may hold the bus as the result of a bus grant. Refer to the PCI 2.3 specification for the rules by which the PCI controller completes transactions when the timer has expired.
2–0	—	Reserved

13.3.3.11 Header Type Configuration Register

Figure 13-29 shows the read-only header type register, which is hard-wired to 0x00.

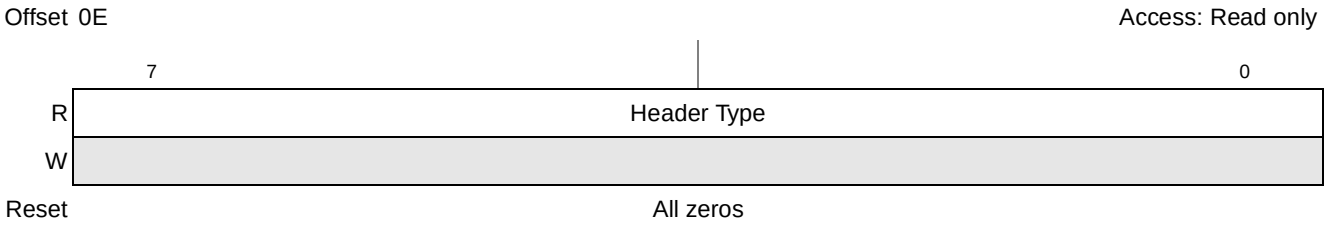


Figure 13-29. Header Type Configuration Register

13.3.3.12 BIST Control Configuration Register

Figure 13-30 shows the read-only BIST control register, which is hard-wired to 0x00.

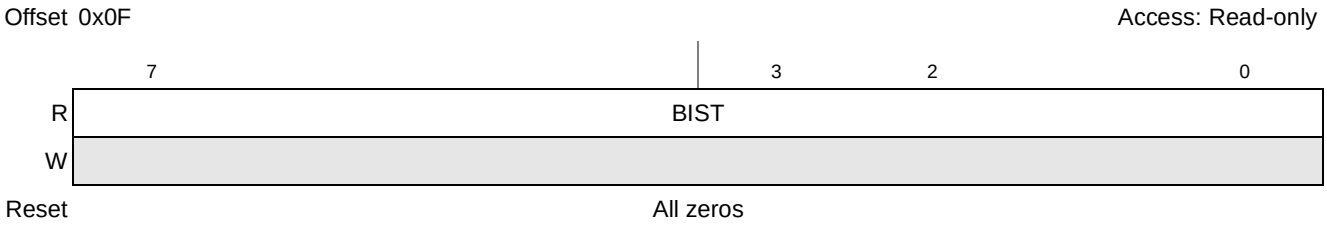


Figure 13-30. BIST Control Configuration Register

13.3.3.13 PIMMR Base Address Configuration Register

Figure 13-31 shows the PIMMR base address register fields.

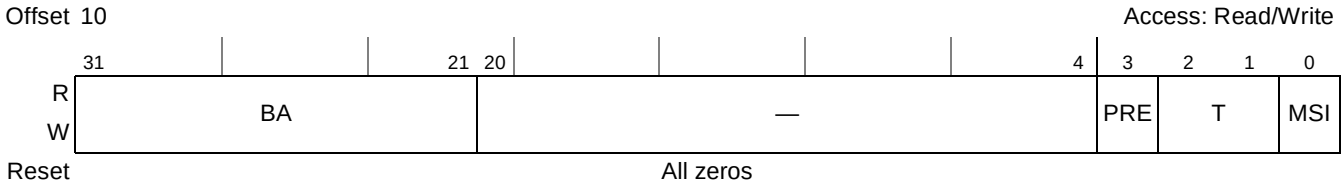


Figure 13-31. PIMMR Base Address Configuration Register

Table 13-33 shows the bit settings of the PIMMR base address register.

Table 13-33. PIMMR Base Address Configuration Register Field Descriptions

Bits	Name	Description
31–21	BA	Base address. Defines the base address for the internal (on-chip) memory-mapped register space. The size of this space is 2 MB.
20–4	—	Reserved
3	PRE	Prefetchable. Hard-wired to 0.
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

13.3.3.14 GPL Base Address Register 0

The GPL base address register 0 is provided to allow access to local memory space. This register is closely tied to PIBAR0 and PIWAR0 in the CSR memory space. A write to GPL base address register 0 also causes a change in the base address bits that are not masked according to the IWS field of PIWAR0 in PIBAR0. Note that this write operation will not change the bits that are masked by the IWS field. For read operation these masked bits will always return zeros.

Figure 13-32 shows the GPL base address register 0 fields.

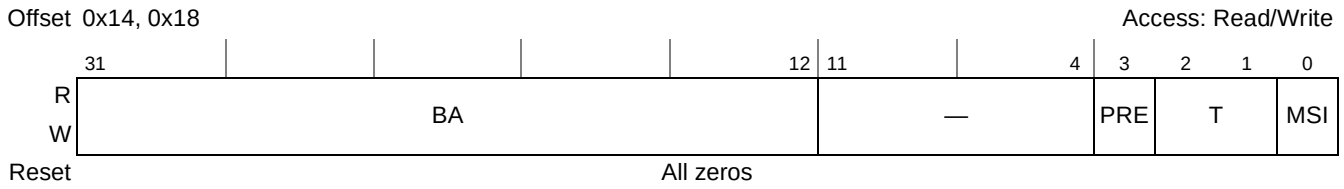


Figure 13-32. GPL Base Address Register 0

Table 13-34 shows the bit settings of the GPL base address register 0.

Table 13-34. GPL Base Address Register 0 Field Descriptions

Bits	Name	Description
31–12	BA	Base address. Defines the base address for the inbound window. Bits 11–4 are hard-wired to 0 because the minimum window size is 4 Kbytes.
3	PRE	Prefetchable. This bit is read-only and contains the value of the PF bit in PIWAR0.
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

13.3.3.15 GPL Base Address Registers 1–2

The general purpose local access base address registers are provided to allow access to local memory space. These registers are closely tied to PIBAR*n* and PIWAR*n* in the CSR memory space. A write to a GPL base address register also causes a change in the base address bits that are not masked according to the IWS field of PIWAR*n* in the corresponding PIBAR*n*. Note that this write operation will not change the bits that are masked by the IWS field. For read operations, these masked bits always return zeros. Figure 13-33 shows the GPL base address register 1–2 fields.

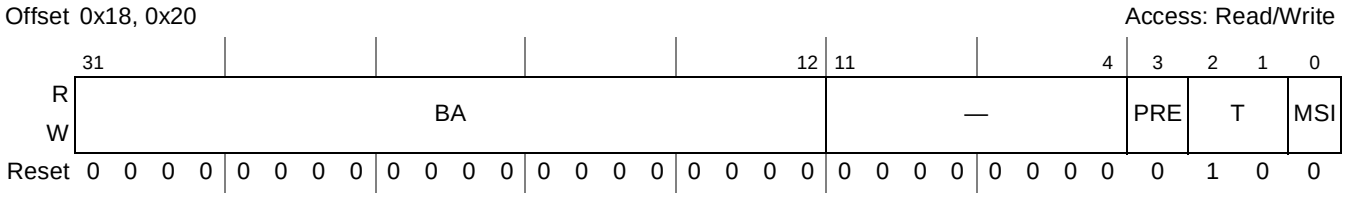


Figure 13-33. GPL Base Address Registers 1–2

Table 13-35 shows the bit settings of the GPL base address register 1–2.

Table 13-35. GPL Base Address Register 1,2 Field Descriptions

Bits	Name	Description
31–12	BA	Base address. Defines the two portion of the base address for the inbound window. Bits 11–4 are hard-wired to 0 because the minimum window size is 4 Kbytes.
11–4	—	Reserved
3	PRE	Prefetchable. This bit is read-only and contains the value of the PF bit in PIWAR*n*.
2–1	T	Type. Hard-wired to 10.
0	MSI	Memory space indicator. Hard-wired to 0.

13.3.3.16 GPL Extended Base Address Registers 1–2

Two general-purpose local access base address registers are provided to allow access to local memory space. These registers are closely tied to PIBAR*n*, PIEBAR*n*, and PIWAR*n* in the CSR memory space. A write to a GPL extended base address register also causes a change in the base address bits that are not masked according to the IWS field of PIWARx in the corresponding PIBAR*n*/PIEBAR*n*. Note that this

write operation does not change bits that are masked by the IWS field. For read operations these masked bits will always return zeros.

Figure 13-34 shows the GPL extended base address registers 1–2 fields.

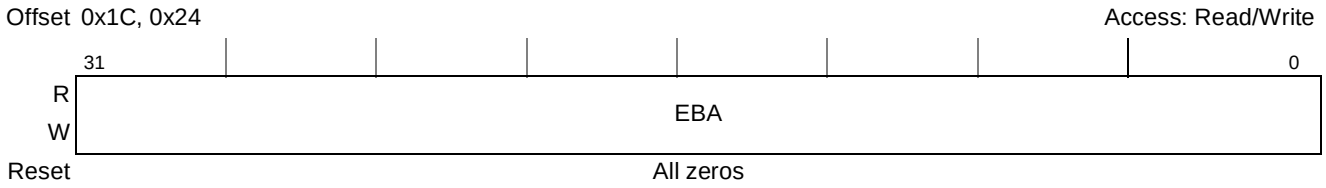


Figure 13-34. GPL Extended Base Address Registers 1–2

Table 13-36 shows the bit settings of the GPL extended base address register 1–2.

Table 13-36. GPL Extended Base Address Registers 1–2 Field Descriptions

Bits	Name	Description
31–0	EBA	Extended base address. Defines the high portion of the base address for the inbound window.

13.3.3.17 Subsystem Vendor ID Configuration Register

Figure 13-35 shows the subsystem vendor ID fields. The subsystem vendor ID configuration register is read-only from the PCI bus, but it can be programmed from the CSB.

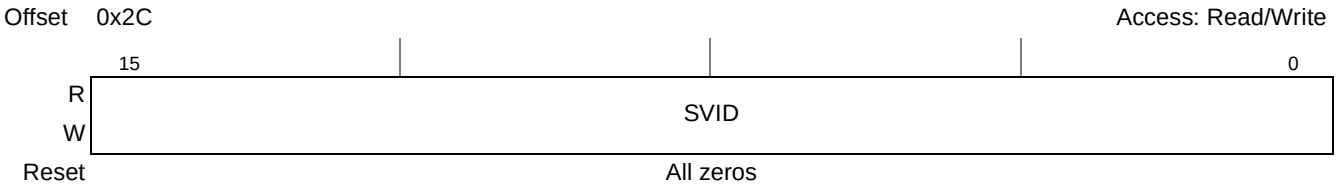


Figure 13-35. Subsystem Vendor ID Configuration Register

Table 13-37 shows the bit settings of the subsystem vendor ID configuration register.

Table 13-37. Subsystem Vendor ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	SVID	Subsystem vendor ID. Identifies the manufacturer of the board or subsystem that contains this device.

13.3.3.18 Subsystem Device ID Configuration Register

Figure 13-36 shows the subsystem device configuration register ID fields. The subsystem device ID configuration register is read-only from the PCI bus, but it can be programmed from the CSB.

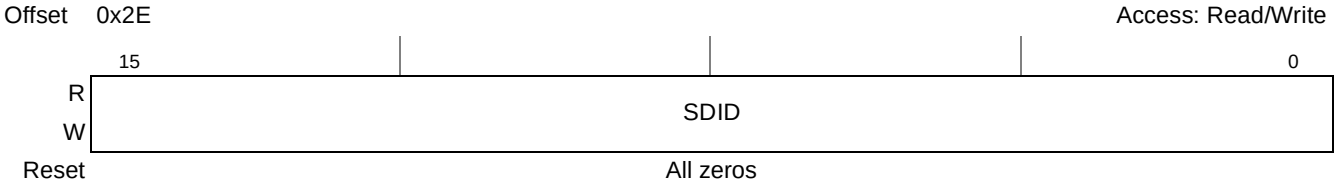


Figure 13-36. Subsystem Device ID Configuration Register

Table 13-38 shows the bit settings of the subsystem device ID configuration register.

Table 13-38. Subsystem Device ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	SDID	Subsystem device ID. This field identifies the board or subsystem that contains this device.

13.3.3.19 Capabilities Pointer Configuration Register

The capabilities pointer register specifies the byte offset in the PCI configuration space that contains the first item in the capabilities list. Figure 13-37 shows the capabilities pointer configuration register fields.

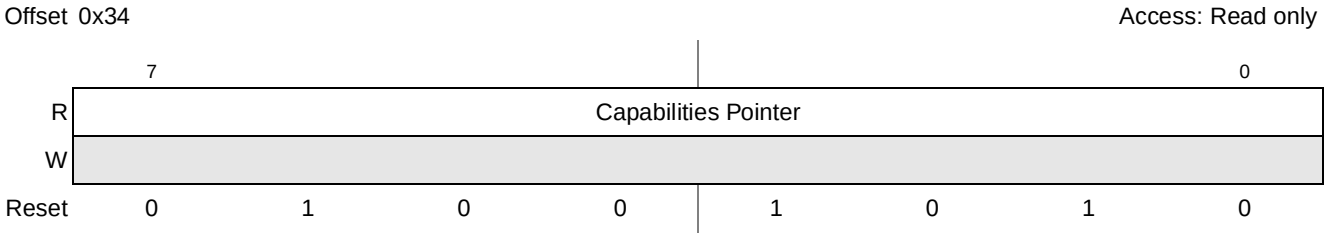


Figure 13-37. Capabilities Pointer Configuration Register

13.3.3.20 Interrupt Line Configuration Register

Figure 13-38 shows the interrupt line configuration register fields.

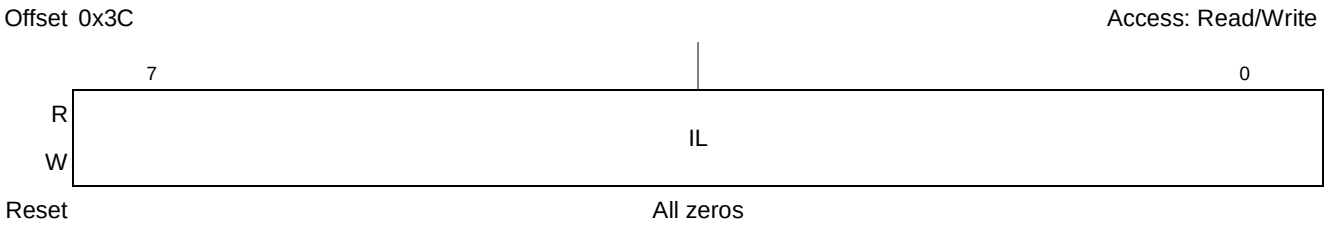


Figure 13-38. Interrupt Line Configuration Register

Table 13-39 shows the bit settings of the interrupt line configuration register.

Table 13-39. Interrupt Line Configuration Register Field Descriptions

Bits	Name	Description
7–0	IL	Interrupt line. Used to communicate interrupt line routing information. The value has no effect on the operation of the PCI controller.

13.3.3.21 Interrupt Pin Configuration Register

The interrupt pin configuration register tells which interrupt pin is used (0x01 means PCI_INTA). Figure 13-39 shows the interrupt pin configuration register fields.

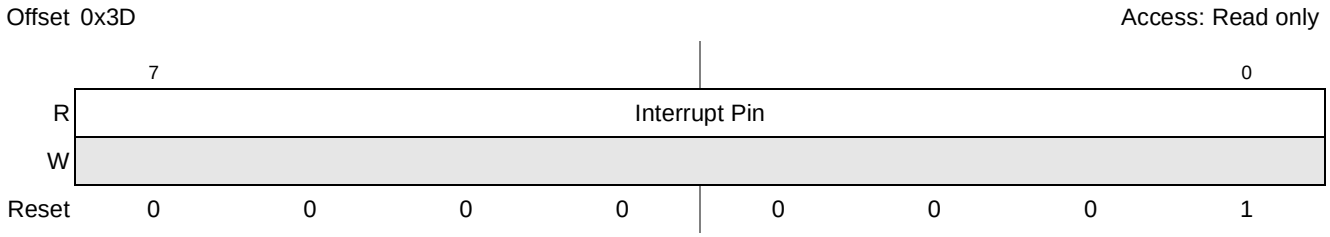


Figure 13-39. Interrupt Pin Register

13.3.3.22 Minimum Grant Configuration Register

Figure 13-40 shows the minimum grant configuration register fields.

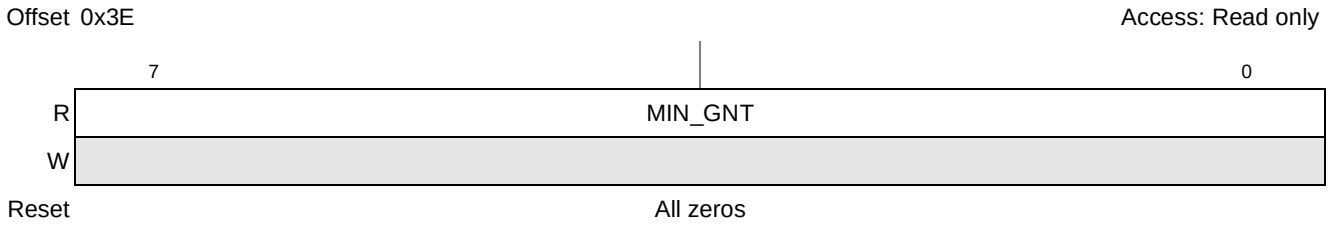


Figure 13-40. Minimum Grant Configuration Register

13.3.3.23 Maximum Latency Configuration Register

Figure 13-41 shows the maximum latency configuration register fields.

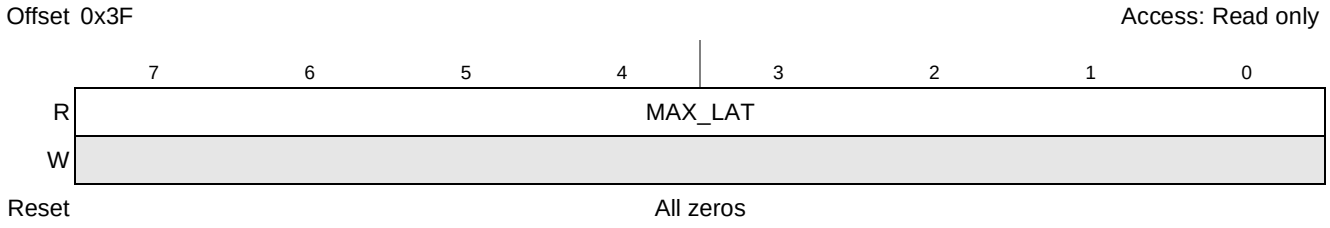


Figure 13-41. Maximum Latency Configuration Register

13.3.3.24 PCI Function Configuration Register

Figure 13-42 shows the PCI function configuration register fields.

Offset 0x44 Access: Read/Write

	15								6	5	4	3	2	1	0
R W	—									CFG_ LOCK	—		TLTD	MLTD	HA
Reset	0	0	0	0	0	0	0	0	0	0	1	0	cfg	0	0	cfg

Figure 13-42. PCI Function Configuration Register

Table 13-40 shows the bit settings of the PCI function configuration register.

Table 13-40. PCI Function Configuration Register Field Descriptions

Bits	Name	Description
15–6	—	Reserved
5	CFG_LOCK	Configuration lock. Controls access to the PCI configuration space from the PCI port. In host mode the PCI configuration space is always inaccessible, so this bit is not used. Normally, this bit will be cleared in agent mode once the configuration of the PCI controller is complete to allow an external host to access the PCI configuration space. 0 Access to the configuration spaces is permitted. 1 Any inbound PCI access to the PCI configuration space is retried. SeeSection 4.3.1.1, “Reset Configuration Word Source,” for more information on reset configuration.
3–4	—	Reserved
2	TLTD	Target latency timeout disable. Determines whether the PCI controller, while acting as a PCI target, times out when the first data phase of a transaction has not completed in 16 PCI cycles. 0 Target latency timeout enabled. 1 Target latency timeout disabled.
1	MLTD	Master latency timer disable. Determines whether the PCI controller, while acting as a PCI master, terminates a transaction upon the expiration of the master latency timer. 0 Master latency timer enabled. 1 Master latency timer disabled.
0	HA	Host/Agent. Indicates whether the PCI controller is in host mode or agent mode. It provides the value of the PCI_HOST—PCI host configuration bit is sampled at the end of the reset sequence. 0 Host mode 1 Agent mode

13.3.3.25 PCI Arbiter Control Register (PCIACR)

Figure 13-43 shows the PCI arbiter control register (PCIACR) fields.

Offset 0x46 Access: Read/Write

	15	14	13	12	11				7	6	5	4	3			1	0
R W	AD	PM	—	PMBD	—					PRI0	PRI1	PRI2	—				MPRI
Reset	cfg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 13-43. PCI Arbiter Control Register (PCIACR)

Table 13-41 shows the bit settings of the PCIACR.

Table 13-41. PCI Arbiter Control Register (PCIACR) Field Descriptions

Bits	Name	Description
15	AD	Arbiter disable. Indicates whether the PCI controller functions as the arbiter for the PCI bus. It provides the value of the PCI arbiter enable configuration bit as sampled at the end of the reset sequence. SeeChapter 4, "Reset, Clocking, and Initialization," for more information on reset configuration. 0 Arbiter enabled 1 Arbiter disabled
14	PM	Parking mode. Controls which device receives a bus grant when there are no outstanding bus requests and the bus is idle. 0 The bus is parked with the last device to use the bus. 1 The bus is parked with the PCI controller.
13	—	Reserved
12	PBMD	PCI broken master disable. Determines whether the PCI controller ignores the bus requests of an initiator that requests the bus for an excessive period without using it. 0 An initiator that requests the bus and receives the grant must begin using the bus within 16 PCI clock periods after the bus becomes idle or its request is subsequently ignored. 1 No requests are ignored.
11–7	—	Reserved
6–4	PRI*n*	Priority level for master *n*. When the PCI controller functions as the arbiter for the PCI bus, each PRIn bit determines the arbitration priority level for the PCI master connected to the REQ*n*/GNT*n* pair. 0 Low priority 1 High priority
3–1	—	Reserved
0	MPRI	My priority. When the PCI controller functions as the arbiter for the PCI bus, this bit determines the arbitration priority level for the PCI controller when it acts as a PCI master. 0 Low priority 1 High priority

13.3.3.26 Hot Swap Register Block

Figure 13-44 shows the hot swap register block fields.

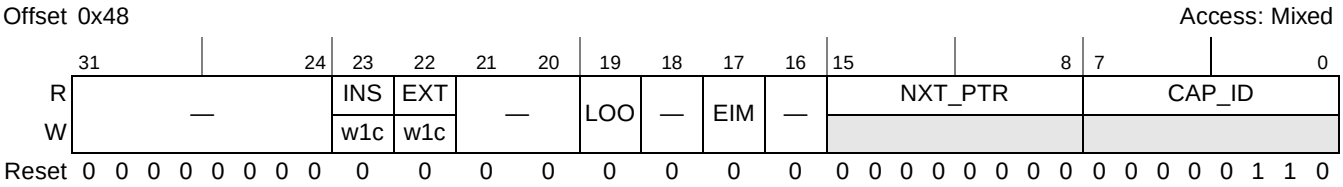


Figure 13-44. Hot Swap Register Block

Table 13-42 shows the bit settings of the Hot Swap register block.

Table 13-42. Hot Swap Register Block Field Descriptions

Bits	Name	Description
31–24	—	Reserved
23	INS	Insertion status. Indicates that a card has been inserted. Write 1 to clear this bit.
22	EXT	Extraction status. Indicates that a card has been extracted. Write 1 to clear this bit.
21–20	—	Reserved
19	LOO	LED On/Off. Controls the LED when the hardware is in state H2 0 LED off 1 LED on
18	—	Reserved
17	EIM	ENUM mask. This bit masks the CPCI_HS_ENUM input. 0 Enabled 1 Masked
16	—	Reserved
15–8	NXT_PTR	Next pointer—hardwired to 0x00 to indicate that this is the last item in the capabilities list.
7–0	CAP_ID	Capability ID for hot swap (hardwired to 0x06)

13.4 Functional Description

The following sections discuss the operation of the PCI controller.

13.4.1 PCI Bus Arbitration

The PCI bus arbitration approach is access-based. Bus masters must arbitrate for each access performed on the bus. PCI uses a central arbitration scheme where each master has its own unique request ($\overline{\text{REQ}}n$) output and grant ($\overline{\text{GNT}}n$) input signal. A simple request-grant handshake is used to gain access to the bus. Arbitration for the bus occurs during the previous access so that no PCI bus cycles are consumed waiting for arbitration (except when the bus is idle).

The PCI internal arbiter supports three external masters (besides the PCI controller itself) by using the $\overline{\text{REQ}}$ signals and generating the $\overline{\text{GNT}}$ signals.

During reset, the PCI controller samples the reset configuration bit (and programs the PCI_ARB_DIS bit accordingly) to determine if the arbiter is enabled or disabled. SeeChapter 4, “Reset, Clocking, and Initialization,” for more information. The arbiter can also be enabled or disabled by directly programming the PCI_ARB_DIS bit in the arbiter configuration register (see Section 13.3.3.25, “PCI Arbiter Control Register (PCIACR),” for more information). However, it is recommended to use the reset configuration bit to set the arbiter state because the arbiter state controls the direction of $\overline{\text{REQ0}}$ and $\overline{\text{GNT0}}$.

If the arbiter is disabled, the PCI controller uses $\overline{\text{REQ0}}$ to issue requests to an external arbiter, and uses $\overline{\text{GNT0}}$ to receive grants from the external arbiter.

13.4.1.1 Bus Parking

When no devices are requesting the bus, the bus is granted, or parked, for a specified device to prevent the AD, PCI_C/$\overline{BE}$ and PCI_PAR signals from floating. The PCI controller can be configured to either park on itself or park on the last master to use the bus (see Section 13.3.3.25, "PCI Arbiter Control Register (PCIACR)," for more information).

13.4.1.2 Arbitration Algorithm

The round-robin arbitration algorithm has two priority levels. Each of the external PCI bus masters, plus the PCI controller, are assigned either a high or a low priority level, as programmed in the arbiter configuration register (see Section 13.3.3.25, "PCI Arbiter Control Register (PCIACR).") Within each priority group (high or low), the bus grant is given to the next requesting device in numerical order, with the PCI controller itself positioned before device 0. $\overline{GNTn}$ is asserted for device *n* as soon as the previously granted device begins a transaction. Conceptually, the lowest priority device at any given time is the current bus master and the highest priority device is the next one to follow the current master. This is considered to be a fair algorithm because a given device cannot prevent other devices from having access to the bus—a given device automatically becomes the lowest priority device as soon as it begins to use the bus. If a master is not requesting the bus, the transaction slot is given to the next requesting device within the priority group.

The grant given to one device may be taken away and whenever a higher priority device asserts its request. If the bus is idle when a new device is to receive a grant, no device receives a grant for one clock; in the next clock, the new winner of the arbitration receives a grant. This operation allows for a turnaround clock when a device is using address stepping or when the bus is parked.

The low priority group collectively receives one bus transaction request slot in the high priority group. Therefore, if there are N high-priority devices, each high-priority device is guaranteed to get at least one of (N+1) bus transactions, and the M low priority devices are guaranteed to each get at least one of (N+1) x M bus transactions, with one of the low-priority devices receiving the grant in one of (N+1) bus transactions. If all devices are programmed to the same priority level or if there is only one device at the low priority, the algorithm provides each device an equal number of bus grants in a round-robin sequence.

An arbitration example with three masters in the high priority group and two in the low priority group is shown in Figure 13-45. Noting that one position in the high priority group is actually a place-holder for the low priority group, it can be seen that each high priority initiator is guaranteed at least 1 out of 3 transaction slots, and each low priority initiator is guaranteed at least 1 out of 6 slots. Assuming all devices are requesting the bus, the grant sequence (with device 1 being the current master) is as follows: 0, 2, the PCI controller, 0, 2, 1, 0, 2, the PCI controller, and so on. If, for example, device 2 is not requesting the bus, the grant sequence becomes 0, the PCI controller, 0, 1, 0, the PCI controller, and so on. If device 2 now requests the bus at a point in the sequence when device 0 is conducting a transaction and the PCI controller is the next grant, then the PCI controller's grant is removed, and the higher-priority device 2 is awarded the next grant.

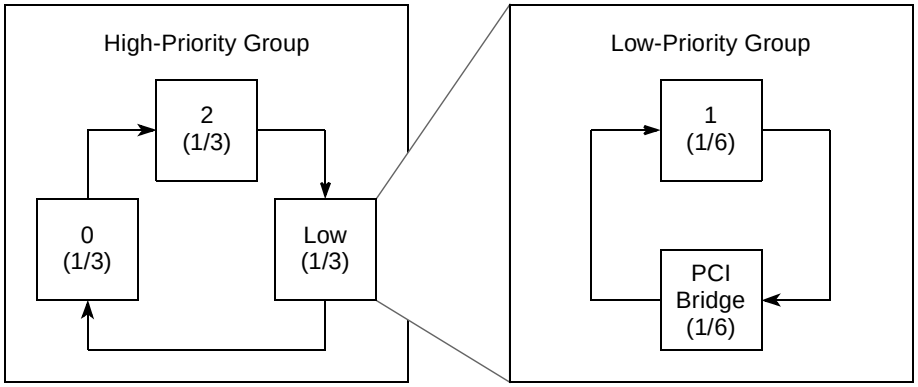


Figure 13-45. PCI Arbitration Example

13.4.1.3 Broken Master Lock-Out

The broken master feature allows the arbiter to lock out any masters that are broken or ill-behaved. This feature is controlled by programming the PCI arbiter control register. When the broken master feature is enabled, a granted device that does not assert $\overline{\text{PCI\_FRAME}}$ within 16 PCI clock cycles after the bus is idle, has its grant removed and subsequent requests are ignored until its $\overline{\text{REQ}}$ is negated for at least one clock cycle. This prevents ill-behaved masters from monopolizing the bus. When the broken master feature is disabled, a device that requests the bus and receives a grant never loses its grant until and unless it begins a transaction or negates its $\overline{\text{REQ}}$ signal. Note that disabling the broken master feature is not recommended.

13.4.1.4 Master Latency Timer

The PCI controller implements the master latency timer register (see Section 13.3.3.10, “Latency Timer Configuration Register”) to prevent itself from monopolizing the bus. When the master latency timer expires, the PCI controller checks the state of its $\overline{\text{PCI\_GNT}}$ signals. If the $\overline{\text{PCI\_GNT}}$ signal is not asserted, the PCI controller completes one more data phase and relinquishes the bus. The master latency timer can be disabled if needed (see Section 13.3.3.24, “PCI Function Configuration Register,” for more information).

13.4.2 Bus Commands

PCI bus commands indicate the type of transaction occurring on the bus. These commands are encoded on PCI_C/$\overline{BE}$[3:0] during the address phase of the transaction. PCI bus commands are described in Table 13-43.

Table 13-43. PCI Command Definitions

PCI_C/$\overline{BE}$[3:0]	Command Type	Supported as:		Definition
		Initiator	Target	
0b0000	Interrupt acknowledge	Yes	No	A read implicitly addressed to the system interrupt controller. The size of the vector to be returned is indicated on the byte enables after the address phase.
0b0001	Special cycle	Yes	No	Provides a simple message broadcast mechanism. See Section 13.4.4.6, “Special Cycle Command,” for more information.
0b0010	I/O read	Yes	No	Accesses agents mapped in I/O address space.
0b0011	I/O write	Yes	No	Accesses agents mapped in I/O address space.
0b010x	—	—	—	Reserved. No response occurs.
0b0110	Memory read	Yes	Yes	Accesses agents mapped in memory address space. A read from prefetchable space, when seen as a target, fetches a cache line of data (32 bytes) from the starting address, even though all 32 bytes may not actually be sent to the initiator.
0b0111	Memory write	Yes	Yes	Accesses agents mapped in memory address space. Note that for inbound writes less than 4-bytes, the PCI controller splits the transaction into single byte writes to the target. Thus, the PCI interface cannot be used to perform single beat writes to 16-bit devices on the local peripheral interfaces.
0b100x	—	—	—	Reserved. No response occurs.
0b1010	Configuration read	Yes	Yes	Accesses the configuration space of each agent. An agent is selected when its IDSEL signal is asserted. See Section 13.4.4.4, “Host Mode Configuration Access,” for more information on configuration accesses. As a target, a configuration read is only accepted if the PCI controller is configured to be in agent mode.
0b1011	Configuration write	Yes	Yes	Accesses the configuration space of each agent. An agent is selected when its IDSEL signal is asserted. See Section 13.4.4.4, “Host Mode Configuration Access,” for more information. As a target, a configuration write is only accepted if the PCI controller is configured to be in agent mode.
0b1100	Memory read multiple	Yes	Yes	Causes a prefetch of the next cache line.
0b1101	Dual address cycle	No	Yes	Transfers an 8-byte address to devices.
0b1110	Memory read line	Yes	Yes	Indicates that the initiator intends to transfer an entire cache line of data.
0b1111	Memory write and invalidate	No	Yes	Indicates that the initiator will transfer an entire cache line of data, and if PCI has any cacheable memory, this line needs to be invalidated.

13.4.3 PCI Protocol Fundamentals

The bus transfer mechanism on the PCI bus is called a burst. A burst is comprised of an address phase and one or more data phases.

All signals are sampled on the rising edge of the PCI clock. Each signal has a setup and hold window with respect to the rising clock edge, in which transitions are not allowed. Outside this aperture, signal values or transitions have no significance.

13.4.3.1 Basic Transfer Control

PCI data transfers are controlled by the following signals:

- $\overline{\text{PCI\_FRAME}}$ is driven by an initiator to indicate the beginning and end of a transaction.
- $\overline{\text{PCI\_IRDY}}$ (initiator ready) is driven by an initiator, allowing it to force wait cycles.
- $\overline{\text{PCI\_TRDY}}$ (target ready) is driven by a target, allowing it to force wait cycles.

The bus is idle when both $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated. The first clock cycle in which $\overline{\text{PCI\_FRAME}}$ is asserted indicates the beginning of the address phase. The address and the bus command code are transferred in that cycle. The next cycle ends the address phase and begins the data phase.

During the data phase, data is transferred in each cycle that both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted. Once the PCI controller, as an initiator, has asserted $\overline{\text{PCI\_IRDY}}$, it does not change $\overline{\text{PCI\_IRDY}}$ or $\overline{\text{PCI\_FRAME}}$ until the current data phase completes, regardless of the state of $\overline{\text{PCI\_TRDY}}$. Once the PCI controller, as a target, has asserted $\overline{\text{PCI\_TRDY}}$ or $\overline{\text{PCI\_STOP}}$ it does not change $\overline{\text{PCI\_DEVSEL}}$, $\overline{\text{PCI\_TRDY}}$, or $\overline{\text{PCI\_STOP}}$ until the current data phase completes.

When the PCI controller (as a master) intends to complete only one more data transfer, $\overline{\text{PCI\_FRAME}}$ is negated and $\overline{\text{PCI\_IRDY}}$ is asserted (or kept asserted) indicating the initiator is ready. After the target indicates it is ready ($\overline{\text{PCI\_TRDY}}$ asserted) the bus returns to the idle state.

13.4.3.2 Addressing

The PCI specification defines three physical address spaces—memory, I/O, and configuration. The memory and I/O address spaces are standard for all systems. The configuration address space supports the PCI hardware configuration. Each PCI device decodes the address for each PCI transaction with each agent responsible for its own address decode.

The information contained in the two lower address bits (AD1 and AD0) depends on the address space. In the I/O address space, all 32 address/data lines provide the full byte address. AD[1:0] are used for the generation of $\overline{\text{PCI\_DEVSEL}}$ and indicate the least significant valid byte involved in the transfer. Once a target has claimed an I/O access, it first determines if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range, the entire access should not be completed; that is, the target should not transfer any data and should terminate the transaction with a target-abort operation. See Section 13.4.3.6, “Bus Transactions,” for more information.

In the configuration address space, accesses are decoded to a 4-byte address using AD[7:2]. An agent determines if it is the target of the access when a configuration command is decoded, IDSEL is asserted, and AD[1:0] are 0b00; otherwise, the agent ignores the current transaction. The PCI controller determines

a configuration access is for a device on the PCI bus by decoding a configuration command. When in agent mode, the PCI controller responds to host-generated PCI configuration cycles when its IDSEL is asserted during a configuration cycle.

For memory accesses, the address is decoded using AD[31:2]; thereafter, the address is incremented internally by 4 bytes until the end of the burst transfer. Another initiator in a memory access should drive 0b00 on AD[1:0] during the address phase to indicate a linear incrementing burst order. The PCI controller checks AD[1:0] during a memory command access and provides the linear incrementing burst order. On reads, if AD[1:0] is 0b10, which represents a cache line wrap, the PCI controller linearly increments the burst order starting at the critical 64-bit address, wraps at the end of the cache line, and disconnects after reading one cache line. If AD[1:0] is 0bx1 (a reserved encoding) and the PCI_C/$\overline{\text{BE}}$[3:0] signals indicate a memory transaction, it executes a target disconnect after the first data phase is completed. Note that AD[1:0] are included in parity calculations.

13.4.3.3 Device Selection

As a target, the PCI controller drives $\overline{\text{PCI\_DEVSEL}}$ one clock following the address phase as indicated in the configuration space status register; see Section 13.3.3.4, “PCI Status Configuration Register,” for more information. The PCI controller as a target qualifies the address/data lines with $\overline{\text{PCI\_FRAME}}$ before asserting $\overline{\text{PCI\_DEVSEL}}$. The $\overline{\text{PCI\_DEVSEL}}$ signal is asserted at or before the clock edge at which the PCI controller enables its $\overline{\text{PCI\_TRDY}}$, $\overline{\text{PCI\_STOP}}$, or data (for a read). The $\overline{\text{PCI\_DEVSEL}}$ signal is not negated until $\overline{\text{PCI\_FRAME}}$ is negated, with $\overline{\text{PCI\_IRDY}}$ asserted and either $\overline{\text{PCI\_STOP}}$ or $\overline{\text{PCI\_TRDY}}$ asserted. The exception to this is a target-abort; see Section 13.4.3.8, “Transaction Termination,” for more information.

As an initiator, if the PCI controller does not see the assertion of $\overline{\text{PCI\_DEVSEL}}$ within 4 clocks of $\overline{\text{PCI\_FRAME}}$, it terminates the transaction with a master-abort as described in Section 13.4.3.8, “Transaction Termination,” for more information.

13.4.3.4 Byte Enable Signals

The byte enable signals ($\overline{\text{BE}}$[3:0]) indicate which byte lanes carry valid data. The byte enable signals may enable different bytes for each of the data phases. The byte enable signals are valid on the edge of the clock that starts each data phase and remain valid for the entire data phase.

If the PCI controller, as a target, sees no byte enable signals asserted, it completes the current data phase with no permanent change. This implies that on a read transaction, the PCI controller expects the data not to be changed, and on a write transaction, the data is not stored.

13.4.3.5 Bus Driving and Turnaround

The turnaround-cycle is one clock cycle and is required to avoid contention. This cycle occurs at different times for different signals. $\overline{\text{PCI\_IRDY}}$, $\overline{\text{PCI\_TRDY}}$, and $\overline{\text{PCI\_DEVSEL}}$ use the address phase as their turnaround-cycle. $\overline{\text{PCI\_FRAME}}$, PCI_C/$\overline{\text{BE}}$[3:0], and AD[31:0] use the idle cycle between transactions as their turnaround-cycle. (An idle cycle in PCI is when both $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated).

Byte lanes not involved in the current data transfer are driven to a stable condition even though the data is not valid.

13.4.3.6 Bus Transactions

The timing diagrams in this section show the relationship of significant signals involved in bus transactions.

Note the following conventions:

- When a signal is drawn as a solid line, it is actively being driven by the current initiator or target.
- When a signal is drawn as a dashed line, no agent is actively driving it.
- Signals released to high impedance with slashes between the two rails have indeterminate values.
- The terms ‘edge’ and ‘clock edge’ refer to the rising edge of the clock.
- The terms ‘asserted’ and ‘negated’ refer to the globally visible state of the signal on the clock edge, and not to signal transitions.
- The symbol  represents a turnaround-cycle.

13.4.3.7 Read and Write Transactions

Both read and write transactions begin with an address phase followed by a data phase. The address phase occurs when PCI_FRAME is asserted for the first time, and the AD[31:0] signals contain a byte address and the PCI_C/BE[3:0] signals contain a bus command. The data phase consists of the actual data transfer and possible wait cycles; the byte enable signals remain actively driven from the first clock of the data phase through the end of the data transfer.

A read transaction starts when PCI_FRAME is asserted for the first time and the PCI_C/BE[3:0] signals indicate a read command. Figure 13-46 shows an example of a single beat read transaction.

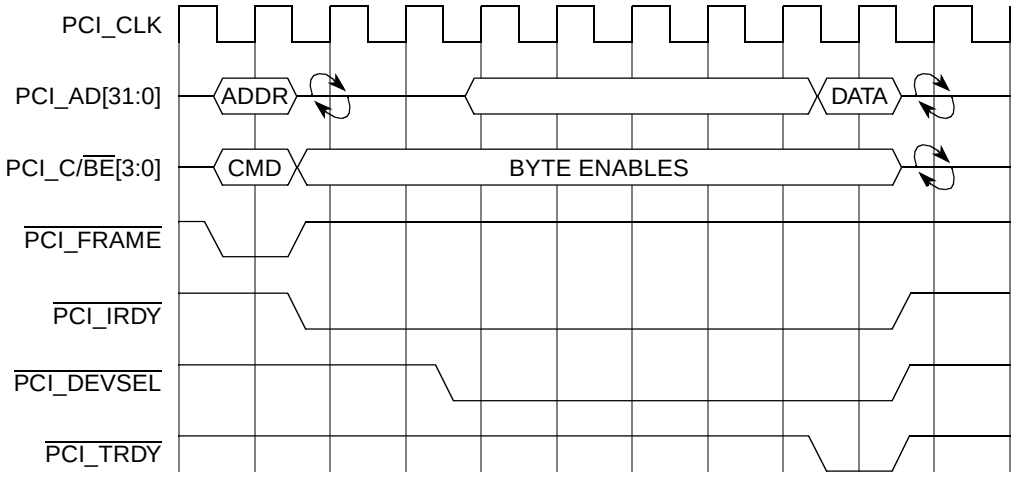


Figure 13-46. Single Beat Read Example

Figure 13-47 shows an example of a burst read transaction.

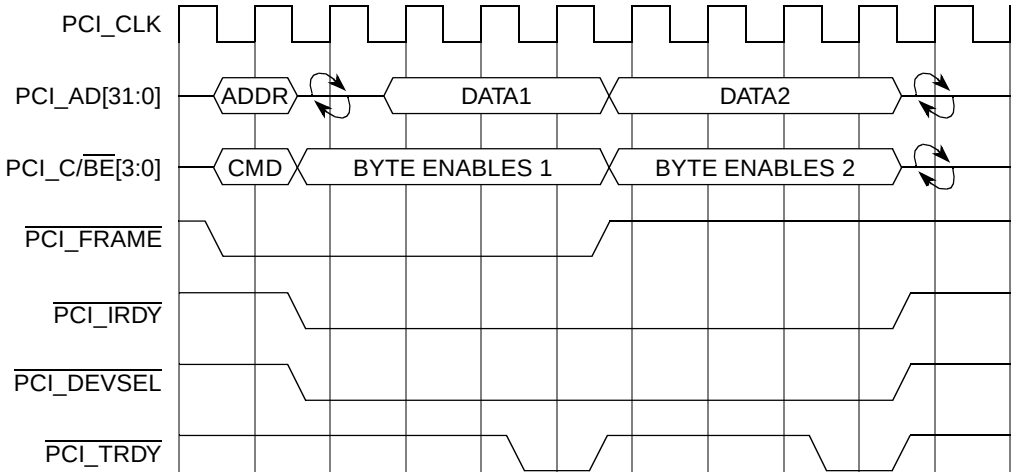


Figure 13-47. Burst Read Example

During the turnaround-cycle following the address phase, the PCI_C/BE[3:0] signals indicate which byte lanes are involved in the data phase. The turnaround-cycle must be enforced by the target with the PCI_TRDY signal if using fast PCI_DEVSEL assertion. The earliest the target can provide valid data is one cycle after the turnaround cycle. The target must drive the AD[31:0] signals when PCI_DEVSEL is asserted except during the turnaround cycle.

The data phase completes when data is transferred, which occurs when both PCI_IRDY and PCI_TRDY are asserted on the same clock edge. When either is negated, a wait cycle is inserted and no data is transferred. To indicate the last data phase PCI_IRDY must be asserted when PCI_FRAME is negated.

A write transaction starts when PCI_FRAME is asserted for the first time and the PCI_C/BE[3:0] signals indicate a write command. Figure 13-48 shows an example of a single-beat write transaction.

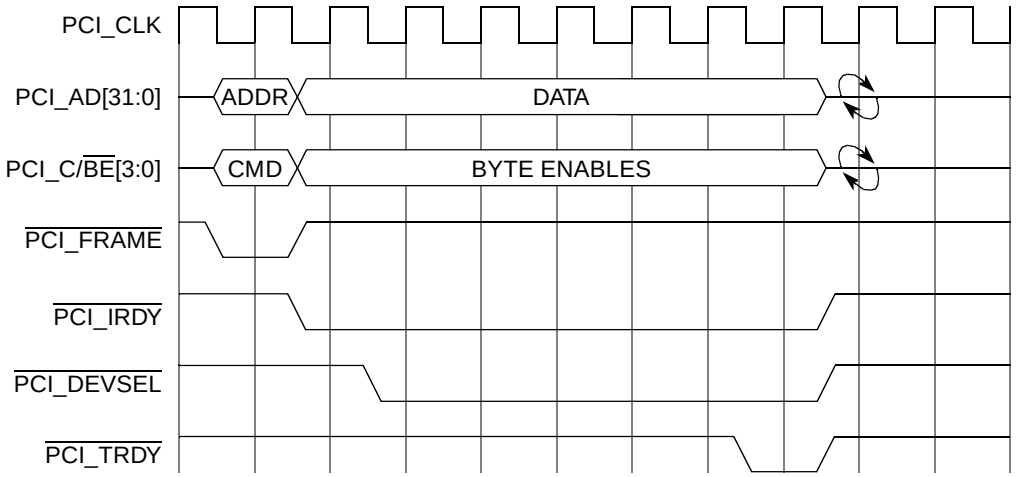


Figure 13-48. Single Beat Write Example

Figure 13-49 shows an example of a burst write transaction.

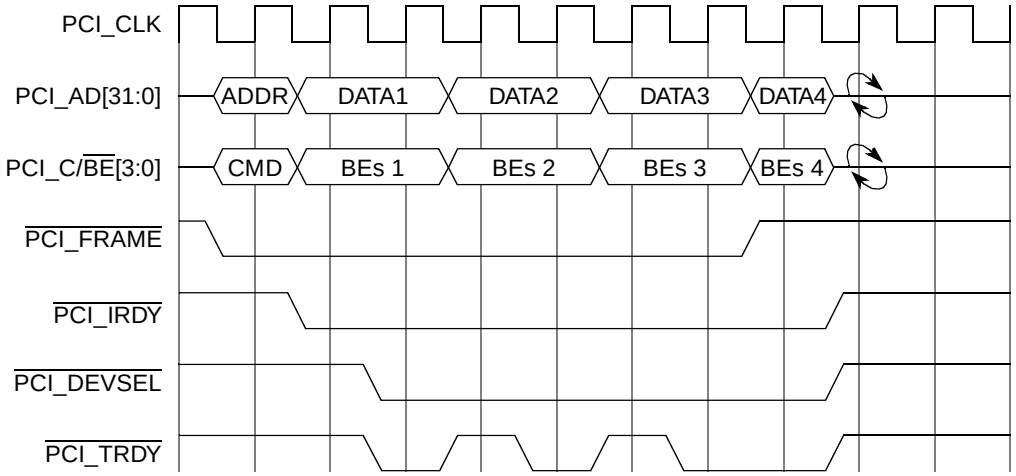


Figure 13-49. Burst Write Example

A write transaction is similar to a read transaction except no turnaround cycle is needed following the address phase because the initiator provides both address and data. Data phases are the same for both read and write transactions.

13.4.3.8 Transaction Termination

The termination of a PCI transaction is orderly and systematic, regardless of the cause of the termination. All transactions end when PCI_FRAME and PCI_IRDY are both negated, indicating the idle cycle.

The PCI controller as an initiator terminates a transaction when PCI_FRAME is negated and PCI_IRDY is asserted. This indicates that the final data phase is in progress. The final data transfer occurs when both PCI_TRDY and PCI_IRDY are asserted. A master-abort is an abnormal case of a master initiated termination. If the PCI controller detects that PCI_DEVSEL has remained negated for more than four clocks after the assertion of PCI_FRAME, it negates PCI_FRAME and then, on the next clock, negates PCI_IRDY. On aborted reads, the PCI controller returns 0xFFFF_FFFF. The data is lost on aborted writes.

When the PCI controller as a target needs to suspend a transaction, it asserts PCI_STOP. Once asserted, PCI_STOP remains asserted until PCI_FRAME is negated. Depending on the circumstances, data may or may not be transferred during the request for termination. If PCI_TRDY and PCI_IRDY are asserted during the assertion of PCI_STOP**,** data is transferred. This type of target-initiated termination is called a disconnect B, shown in Figure 13-50. If PCI_TRDY is asserted when PCI_STOP is asserted but PCI_IRDY is not, PCI_TRDY must remain asserted until PCI_IRDY is asserted and the data is transferred. This is called a disconnect A target-initiated termination, also shown in Figure 13-50. However, if PCI_TRDY is negated when PCI_STOP is asserted, no more data is transferred, and the initiator therefore does not have to wait for a final data transfer (see the retry diagram in Figure 13-48).

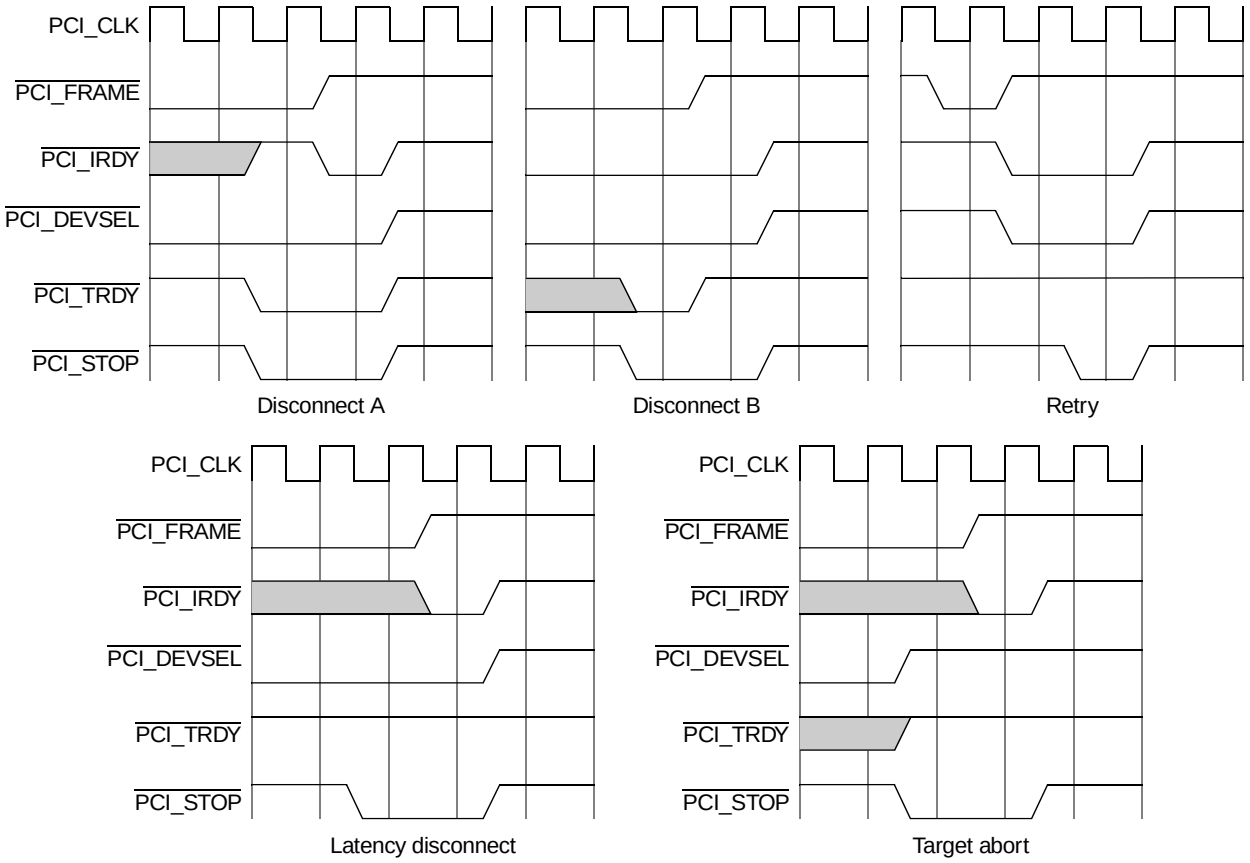


Figure 13-50. Target-Initiated Terminations

Note that when an initiator is terminated by $\overline{\text{PCI\_STOP}}$, it must negate its $\overline{\text{REQ}n}$ signal for a minimum of two PCI clocks (of which one clock is needed for the bus to return to the idle state). If the initiator intends to complete the transaction, it should reassert its $\overline{\text{REQ}n}$ immediately following the two clocks or potential starvation may occur. If the initiator does not intend to complete the transaction, it can assert $\overline{\text{REQ}n}$ whenever it needs to use the PCI bus again.

The PCI controller terminates a transaction in the following cases:

- Eight PCI clock cycles have elapsed between data phases. This is a ‘latency disconnect’ (see Figure 13-48).
- AD[1:0] is 0bx1 **(**a reserved burst ordering encoding) during the address phase and one data phase has completed.
- The PCI command is a configuration command and one data phase has completed.
- A streaming transaction crosses a 4-Kbyte page boundary.
- A streaming transaction runs out of I/O sequencer buffer entries.
- A cache line wrap transaction has completed a cache line transfer.

Another target-initiated termination is the retry termination. Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. This can occur because no buffer entries are available in the I/O sequencer, or the sixteen clock latency timer has expired without

transfer of the first data. The target latency timer of the PCI controller can be optionally disabled. See Section 13.3.3.24, “PCI Function Configuration Register,” for more information.

When the PCI controller is in host mode it does not respond to any PCI configuration transactions. When the PCI controller is in agent mode and the CFG_LOCK lock bit is set (see Section 13.3.3.24, “PCI Function Configuration Register”) the PCI controller retries all transactions to the PCI configuration space or the internal (on-chip) memory-mapped register space. Note that all retried accesses need to be completed. An example of a retry is shown in Figure 13-48.

Note that because a target can determine whether or not data is transferred (when both PCI_IRDY and PCI_TRDY are asserted), if it wants to do only one more data transfer and then stop, it may assert PCI_TRDY and PCI_STOP at the same time.

Target-abort refers to the abnormal termination that is used when a fatal error has occurred, or when a target will never be able to respond. Target-abort is indicated when PCI_STOP is asserted and PCI_DEVSEL is negated. This indicates that the target requires the transaction to be terminated and does not want the transaction tried again. Note that any transferred data may have been corrupted.

The PCI controller terminates a transaction with target-abort in the case in which it is the intended target of a read transaction from system memory and the data from memory is corrupt. If the PCI controller is the intended target of a transaction and an address parity error occurs, or a data parity error occurs on a write transaction to system memory, it continues the transaction on the PCI bus but aborts internally. The PCI controller does not target-abort in this case.

If the PCI controller is mastering a transaction that terminates with a target-abort, undefined data is returned on a read and write data is lost. An example of a target-abort is shown in Figure 13-48.

An initiator may retry any target disconnect accesses, except target-abort, at a later time starting with the address of the next non-transferred data. Retry is actually a special case of disconnect where no data transfer occurs at all and the initiator must start the entire transaction over again.

13.4.4 Other Bus Operations

The following sections provide information on additional PCI bus operations.

13.4.4.1 Fast Back-to-Back Transactions

In the two types of fast back-to-back transactions, the first type places the burden of avoiding contention on the initiator while the second places the burden on all potential targets. The PCI controller as a target supports both types of fast back-to-back transactions but does not support them as an initiator. The PCI controller as a target has the fast back-to-back enable bit hardwired to one, that is, enabled.

For the first type (governed by the initiator), the initiator may only run a fast back-to-back transaction to the same target. For the second type, when the PCI controller detects a fast-back-to-back operation and did not drive PCI_DEVSEL in the previous cycle, it delays the assertion of PCI_DEVSEL and PCI_TRDY for one cycle to allow the other target to get off the bus.

13.4.4.2 Dual Address Cycles

The PCI controller supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as a target only. DACs are different from single address cycles (SACs) in that the address phase takes two PCI beats instead of one PCI beat to transfer (64-bit vs. 32-bit addressing). Only PCI memory commands can use DAC cycles; I/O, configuration, interrupt acknowledge, and special cycle command cannot use DAC cycles. The PCI controller supports single-beat and burst DAC transactions.

13.4.4.3 Data Streaming

The PCI controller provides data streaming for PCI transactions to and from prefetchable memory. In other words, when the PCI controller is a target for a PCI initiated transaction, it supplies or accepts multiple cache lines of data without disconnecting. For PCI transactions to non-prefetchable space, the PCI controller disconnects after the first data phase so streaming cannot occur.

For PCI memory reads, streaming is achieved by performing speculative reads from memory in prefetchable space. A block of memory may be marked as prefetchable by setting the PCI configuration registers bit for the inbound address translation (see Section 13.3.2.14, “PCI Inbound Window Attribute Registers (PIWARn),” for more information) in the following cases:

- When reads do not alter the contents of memory (reads have no side effects)
- When reads return all bytes regardless of the byte enable signals
- When writes can be merged without causing errors

For a memory read command or a memory read line command, the PCI controller reads one cache line from memory. If the transaction crosses a cache line boundary, the PCI controller starts the read of a new cache line. For a memory read multiple command, the PCI controller reads two cache lines from memory. When the PCI transaction finishes the read for the first cache line, the PCI controller performs a speculative read of a third cache line. The PCI controller continues this prefetching until the end of the transaction.

For PCI writes to memory, streaming is achieved by buffering the transaction in the space available within the I/O sequencer. This allows PCI memory writes to execute with no wait states.

A disconnect occurs if the PCI controller runs out of buffer space on writes, or the PCI controller cannot supply consecutive data beats for reads within eight PCI bus clocks of each other. A disconnect also occurs if the transaction crosses a 4-Kbyte page boundary.

13.4.4.4 Host Mode Configuration Access

The PCI controller provides two types of configuration accesses to support hierarchical bridges. To access configuration space, a value is written to the CONFIG_ADDR register specifying which PCI bus, which device, and which configuration register to be accessed.

When the PCI controller sees an access that falls inside the 4 bytes beginning at the CONFIG_DATA address, it checks the enable bit, the device number and the bus number in the CONFIG_ADDR register. If the enable bit is set and the device number is not equal to all ones, a configuration cycle translation is performed. When the device number field is equal to all ones, it has a special meaning (see Section 13.4.4.6, “Special Cycle Command,” for more information).

There are two types of translations supported:

- Type 0 translations—For when the device is on the PCI bus connected to the PCI controller.
- Type 1 translations—For when the device is on another bus somewhere behind the PCI controller.

For type 0 translations, the PCI controller decodes the device number field to assert the appropriate IDSEL line and perform a configuration cycle on the PCI bus with AD[1:0] as 0b00. All 21 IDSEL bits are decoded, starting with bit AD11. That is, if the device number field contains 0b01011, AD11 on the PCI bus is set. The IDSEL lines are bit-wise associated with increasing values for the device number such that AD12 corresponds to 0b01100, and so on up to bit 30 as shown in Table 13-41. AD31 is selected with 0b01010. A device number of 0b11111 indicates a special cycle. Device number 0b00000 is used for configuring the PCI controller itself. Bits 10 through 8 are copied to the PCI bus as an encoded value for components which contain multiple functions. Bits 7 through 2 are also copied onto the PCI bus. The PCI controller implements address stepping on configuration cycles so that the target's PCI_IDSEL, which is connected directly to one of the AD lines, reaches a stable value. This means that a valid address and command are driven on the AD and PCI_C/$\overline{\text{BE}}$ lines one cycle before the assertion of $\overline{\text{PCI\_FRAME}}$.

For type 1 translations, the PCI controller copies the contents of the CONFIG_ADDR register directly onto the PCI address/data lines during the address phase of a configuration cycle, with the exception that AD[1-0] contains 0b01 (not 0b00 as in Type 0 translations).

When the PCI controller is configured as a host device, a local master sometimes needs to perform configuration reads from unpopulated PCI slots (as part of the system configuration). To avoid getting a machine check interrupt, the following steps should be taken:

1. Mask the NORSP bit in the error mask register. See Section 13.3.2.9, "PCI Error Control Register (PCI_ECR)."
2. Perform the PCI configuration reads.
3. Clear the NORSP bit in the error status register.
4. Unmask (write 1) the NORSP bit in the error mask register. See Section 13.3.2.3, "PCI Error Enable Register (PCI_EER)."

13.4.4.5 Agent Mode Configuration Access

When the PCI controller is configured as an agent device, it responds to remote host generated PCI configuration accesses to the PCI interface. This is indicated by decoding the configuration command along with the PCI controller's IDSEL being asserted. A remote host can access the 256-byte PCI configuration area and the memory-mapped configuration registers within the PCI controller.

13.4.4.6 Special Cycle Command

A special cycle command contains no explicit destination address but is broadcast to all PCI agents. Each receiving agent must determine whether the message is applicable to itself. No assertion of $\overline{\text{PCI\_DEVSEL}}$ in response to a special cycle command is necessary.

A special cycle command is like any other bus command in that it has an address phase and a data phase. The address phase starts like all other commands with the assertion of $\overline{\text{PCI\_FRAME}}$ and completes when

PCI_FRAME and PCI_IRDY are negated. Special cycles terminate with a master-abort. (In the special cycle case, the received-master-abort bit in the configuration status register is not set.)

The address phase contains no valid information other than the command field. Even though there is no explicit address, the address/data lines are driven to a stable state and parity is generated. During the data phase, the address/data lines contain the message type and an optional data field. The message is encoded on the sixteen least-significant bits (AD[15:0]). The data field is encoded on AD[31:16]. When running a special cycle, the message and data are valid on the first clock PCI_IRDY is asserted.

When the PCI_CONFIG_ADDRESS register is written with a value so that the bus number matches the bridge bus, the device number is all ones, the function number is all ones, and the register number is zero. The next time the PCI_CONFIG_DATA register is accessed, the PCI controller executes either a special cycle or an interrupt acknowledge command. When the PCI_CONFIG_DATA register is written, the PCI controller generates a special cycle encoding on the command/byte enable lines during the address phase and drives the data from the PCI_CONFIG_DATA register onto the address/data lines during the first data phase.

If the bus number field of the PCI_CONFIG_ADDRESS does not match one of the PCI controller bus numbers, the PCI controller passes the write to PCI_CONFIG_DATA through to the PCI bus as a type 1 configuration cycle as it does any other time the bus number field does not match.

Table 13-44. Special Cycle Commands

Address (AD[15–0])	Message Type	Description
0x0000	SHUTDOWN (SLEEP)	Indicates the processor is entering its most power saving mode
0x0001	HALT (DOZE)	Indicates the processor is entering a power save mode where address decoding is still available
0x0002–0xFFFF	—	Reserved for future commands

13.4.4.7 Interrupt Acknowledge

When the PCI_CONFIG_ADDRESS register is written with a value such that the bus number is 0x00, the device number is all ones, the function number is all ones, and the register number is zero, the next time the PCI_CONFIG_DATA register is accessed the PCI controller does either a special cycle command or an interrupt acknowledge command. When the PCI_CONFIG_DATA register is read, the PCI controller generates an interrupt acknowledge command encoding on the command/byte enable lines during the address phase. During the address phase, AD[31:0] do not contain a valid address but are driven with stable data and valid parity (PCI_PAR). During the data phase, the byte enable signals determine which bytes are involved in the transaction. The interrupt vector must be returned when PCI_TRDY is asserted.

An interrupt acknowledge transaction can also be issued on the PCI bus by reading from the PCI_INT_ACK register.

13.4.5 Error Functions

This section describes PCI bus errors.

13.4.5.1 Parity

During valid 32-bit address and data transfers, parity covers all 32 address/data lines and the 4 command/byte enable lines regardless of whether or not all lines carry meaningful information. Byte lanes not actually transferring data are driven with stable (albeit meaningless) data and are included in the parity calculation. During configuration, special cycle or interrupt acknowledge commands, some address lines are not defined but are still driven to stable values and included in the parity calculation.

Even parity is calculated for all PCI operations: the value of PCI_PAR is generated such that the number of ones on PCI_AD[31:0], PCI_C/$\overline{\text{BE}}$[3:0] and PCI_PAR equals an even number. The PCI_PAR signal is driven when the address/data lines are driven and follow the corresponding address or data by one clock.

The PCI controller checks the parity after all valid address phases (the assertion of $\overline{\text{PCI\_FRAME}}$) and for valid data transfers ($\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ asserted) involving the PCI controller. When an address or data parity error is detected, the detected-parity-error bit in the configuration space status register is set (see Section 13.3.3.4, “PCI Status Configuration Register.”)

13.4.5.2 Error Reporting

Except for setting the detected-parity-error bit, all parity error reporting and response is controlled by the parity-error-response bit (see Section 13.3.3.3, “PCI Command Configuration Register,” for more information). If the parity-error-response bit is cleared, the PCI controller completes all transactions regardless of parity errors (address or data). If the bit is set, the PCI controller asserts $\overline{\text{PCI\_PERR}}$ two clocks after the actual data transfer in which a data parity error is detected, and keeps $\overline{\text{PCI\_PERR}}$ asserted for one clock. When acting as an initiator during a read transaction or as a target involved in a write to system memory the PCI controller asserts $\overline{\text{PCI\_PERR}}$.

Figure 13-51 shows the possible assertion points for PCI_PERR if the PCI controller detects a data parity error.

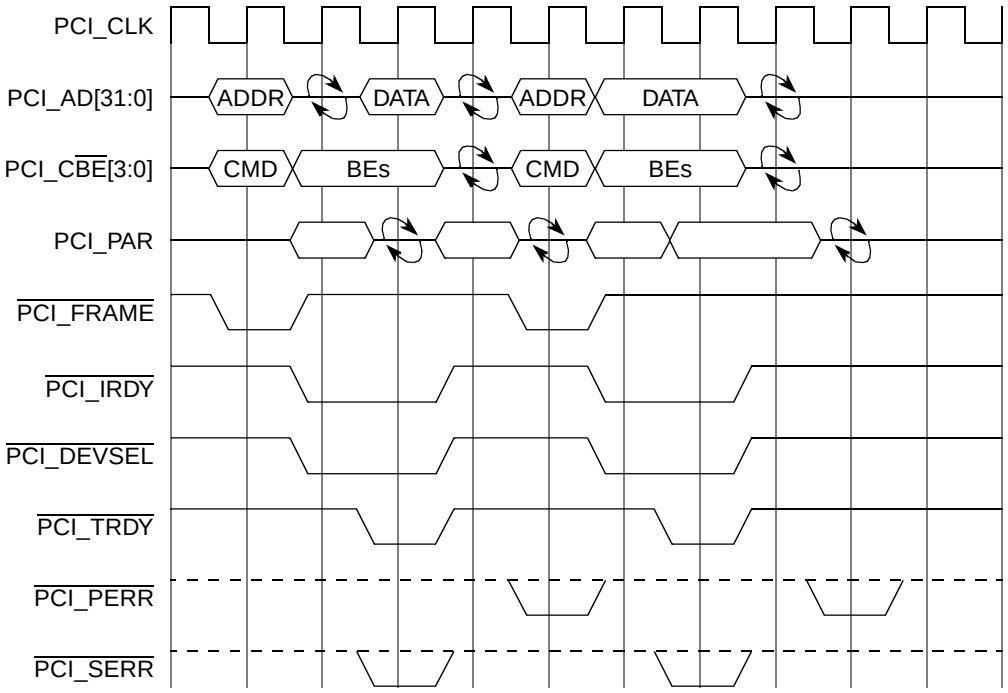


Figure 13-51. PCI Parity Operation

As an initiator, the PCI controller attempts to complete the transaction on the PCI bus if a data parity error is detected and sets the data-parity-reported bit in the configuration space status register. If a data parity error occurs on a read transaction, the PCI controller aborts the transaction internally. As a target, the PCI controller completes the transaction on the PCI bus even if a data parity error occurs. If parity error occurs during a write to system memory, the transaction completes on the PCI bus, but is aborted internally, insuring that potentially corrupt data does not go to memory.

When the PCI controller asserts PCI_SERR, it sets the signaled-system-error bit in the configuration space status register. Additionally, if the error is an address parity error, the parity-error-detected bit is set; reporting an address parity error on PCI_SERR is conditioned on the parity-error-response bit being enabled in the command register. PCI_SERR is asserted when the PCI controller detects an address parity error while acting as a target. The system error is passed to the PCI controller's interrupt processing logic to assert MCP. Figure 13-51 shows where the PCI controller could detect an address parity error and assert PCI_SERR or where the PCI controller, acting as an initiator, checks for the assertion of PCI_SERR signaled by the target detecting an address parity error.

As a target that asserts PCI_SERR on an address parity, the PCI controller completes the transaction on the PCI bus, aborting internally if the transaction is a write to system memory. If PCI_PERR is asserted during a PCI controller write to PCI, the PCI controller attempts to continue the transfer, allowing the target to abort/disconnect if desired. If the PCI controller detects a parity error on a read from PCI, the PCI controller aborts the transaction internally and continues the transfer on the PCI bus, allowing the target to abort/disconnect if desired.

In all cases of parity errors on the PCI bus, regardless of the parity-error-response bit, information about the transaction is logged in the PCI error control capture register, the PCI error address capture register and the PCI error data capture register; $\overline{\text{MCP}}$ is also asserted to the core as an option.

13.4.6 PCI Inbound Address Translation

For inbound transactions (transactions generated by an external master on the PCI bus where the PCI controller responds as a slave device), the PCI controller only responds to PCI addresses within the windows mapped by the PCI inbound base address registers (PIBARs). If there is an address hit in one of the PIBARs, the PCI address is translated from PCI space to local memory space through the associated PCI inbound translation address registers (PITARs). This allows an external master to access local memory. Each PIBAR register is associated with a PITAR and PIWAR which are located in the PCI controller's PCI CSR space. Figure 13-52 shows an example translation window for inbound memory accesses.

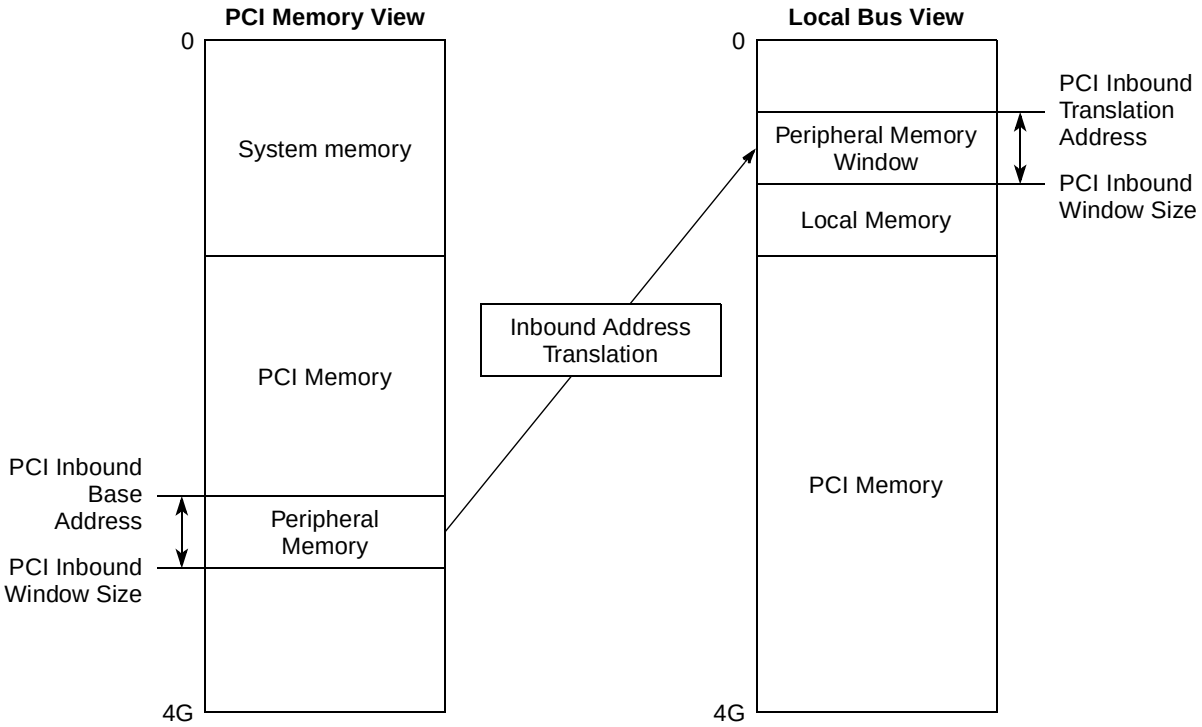


Figure 13-52. Inbound PCI Memory Address Translation

There are three full sets of inbound translation registers, in addition to the PIMMR base address register, allowing four simultaneous translation windows, one to a fixed destination and three programmable. Only two of the programmable windows can be mapped anywhere in the 64-bit PCI address space. Window 0 can only be mapped within the lowest 4-Gbyte space. Software can move the programmable translation base addresses during run-time to access different portions of local memory, but the PCI inbound translation windows may not overlap.

The translation windows are disabled after reset, that is, after reset, the PCI controller does not acknowledge externally mastered transactions on the PCI bus by asserting $\overline{\text{PCI\_DEVSEL}}$ until the inbound translation windows are enabled.

13.4.7 CompactPCI Hot Swap Specification Support

CompactPCI is an open specification supported by the PCI Industrial Computer Manufacturers Group (PICMG) and is intended for embedded applications using PCI. CompactPCI Hot Swap is an extension of the CompactPCI specification and allows the insertion and extraction (or “hot swapping”) of boards without adversely affecting system operation. The hot swap specification defines the following levels of support:

- Hot swap capable
- Hot swap friendly
- Hot swap ready

The PCI controller is hot swap friendly, meaning that it supports the hardware and software connection processes as defined in the hot swap specification. This level of support allows the board and system designers to build full Hot Swap and high availability systems based on the PCI controller as a PCI target device. For details on the hot swap process, refer to the *Hot Swap Specification PICMG 2.1*, R1.0, August 3, 1998.

13.5 Initialization/Application Information

The following sections describe initialization sequences for host and agent modes.

13.5.1 Initialization Sequence for Host Mode

The following sequences must be followed in host mode:

1. Enable PCI output clocks and select desired frequency ratios. See Section 4.4.1, “Clocking in PCI Host Mode.”
2. Wait for at least 1 ms to enable stable clocks into agent devices
3. Deactivate PCI_RESET_OUT signal for PCI. See Table 13-3 for more information on PCI_RESET_OUT signal.
4. Wait for at least 1 ms to enable devices to complete the powerup sequence.
5. Configure PCI internal registers and PCI agents to desired modes of operation

13.5.2 Initialization Sequence for Agent Mode

The following sequences must be followed in agent mode:

1. Optionally initialize subsystem vendor ID/device ID
   a) Initialize PCI inbound window size in PIWAR[1:3] desired window size
   b) Unlock configuration lock in PCI function configuration register

Chapter 14
Security Engine (SEC) 2.2

This chapter describes the functionality of the integrated security engine (SEC 2.2) in the MPC8323E. It addresses the following topics:

NOTE

The MPC8323 and MPC8321 do not support a security engine.

The SEC 2.2 is designed to off-load computationally intensive security functions, such as authentication, and bulk encryption from the processor core of the MPC8323E. It is optimized to process all the algorithms associated with IPSec, SSL/TLS, iSCSI, SRTP, and IEEE 802.11i. The SEC 2.2 is derived from integrated security cores found in other members of the PowerQUICC family, including SEC 1.0, the version implemented in the MPC8272 and SEC 2.0, implemented on the MPC8555.

The security engine’s execution units (EUs) and primary features include the following:

- 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 Rinjdael symmetric key cipher
  - ECB, CBC, CCM, and counter modes
  - 128-, 192-, 256-bit key lengths
- MDEU—Message digest execution unit
  - SHA with 160-, 224-, or 256-bit message digest
  - MD5 with 128-bit message digest
  - HMAC with either algorithm
- One channel, supporting a queue of commands (descriptor pointers)
  - Dynamic assignment of 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. The input and output FIFOs are shared between AESU and DEU; MDEU has its own input FIFO.

- Master/slave logic, with DMA
  - — 32-bit address/64-bit data
  - — DMA blocks can be on any byte boundary
- Scatter/Gather capability
  - — Gather capability enables the SEC 2.2 to concatenate multiple segments of memory when reading input data
  - — Similarly, scatter capability enables the SEC 2.2 to write to multiple segments of memory when writing output data

14.1 SEC 2.2 Architecture Overview

The SEC 2.2 (referred to as SEC in this chapter) can act as a master on the internal system bus to allow the SEC to off-load 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 which 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 the channel with a few short register writes, leaving the SEC to perform reads and writes on system memory to complete the required task.

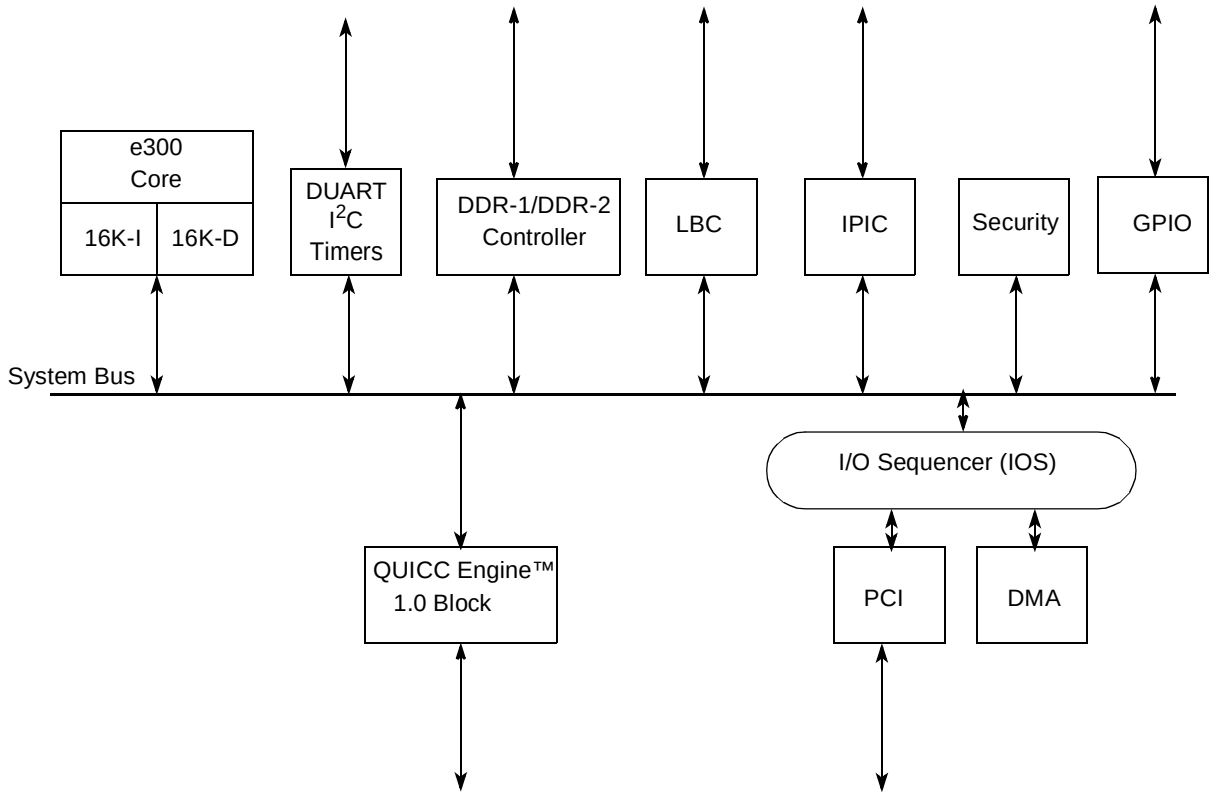


Figure 14-1. SEC Connected to MPC8323E System Bus

Figure 14-2 shows a simplified block diagram of the SEC internal architecture. The controller block is capable of transferring 64-bit words between the bus and any register inside the SEC.

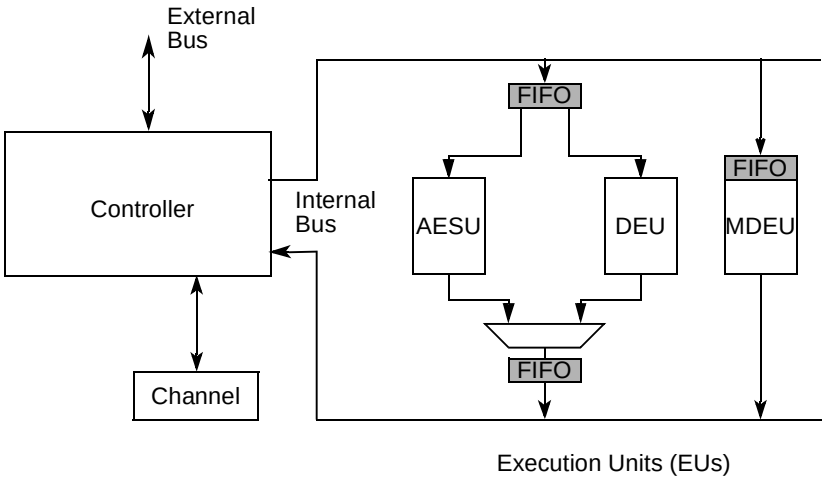


Figure 14-2. SEC Functional Modules

An operation begins when the host writes a descriptor pointer to the fetch FIFO in the SEC channel. 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 execution units needed to perform it. The channel requests the controller to assign the needed execution units. Next the channel requests that the controller fetch the keys, 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 the proper 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 the 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 output FIFO.

14.1.1 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 descriptor is diagrammed in Table 14-1.

Table 14-1. Example Descriptor

Field Name	Value/Type	Description
DPD_DES_CTX_CRYPT	Variable	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 Zeros for fixed length descriptor
Null length Null pointer	Length Pointer	Zeros for fixed length descriptor Zeros for fixed length descriptor

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 that indicates which EUs to use and which modes to set. It also indicates whether 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 non-zero, 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 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 channel can signal done through an interrupt, or by a write-back of to descriptor header in non-SEC memory after processing a descriptor. Either the value written back is identical to that of the header, with the exception that a DONE field is set, or special status fields are written back. That status writeback field can be reserved for descriptors performing ICV checking.

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 descriptors, the channel skips any pointer that has an associated length of zero.

For more information, refer to Section 14.3, “Descriptor Overview.”

14.1.2 Execution Units (EUs)

‘Execution unit’ (EU) is the generic term for a functional block that performs the mathematical manipulations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, SSL/TLS, iSCSI, SRTP, and IEEE 802.11i processing, and can work together to perform high-level cryptographic tasks. The SEC’s execution units are as follows:

- DEU for performing block cipher, symmetric key cryptography using DES and 3DES
- AESU for performing the advanced encryption standard algorithm
- MDEU for performing security hashing using MD-5, SHA-1, SHA-224, or SHA-256

Each EU is described in detail in Section 14.4, “Execution Units.”

14.1.2.1 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 14.4.1, “Data Encryption Standard Execution Unit (DEU).”

14.1.2.2 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the advanced encryption standard algorithm Rinjdael. The AESU executes on 128 bit blocks with a choice of key sizes: 128, 192, or 256 bits.

AESU 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 128-bit input. The AESU operates in ECB, CBC, CTR, and CCM modes.

For more information, refer to Section 14.4.3, “Advanced Encryption Standard Execution Unit (AESU).”

14.1.2.3 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, SHA-224, 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 potential, albeit not probable. The 160-bit hash value is a sufficiently large space such that collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally infeasible to construct two distinct but similar messages that produce the same hash output.

- The MD5 generates a 128-bit hash, and the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the ANSI X9.30-2 and FIPS 180-2 standards.
- SHA-224 is a 224-bit hash function specified by FIPS 180-2 that provides 224 bits of security against collision attacks.
- SHA-256 is a 256-bit hash function specified by FIPS 180-2 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 14.4.2, “Message Digest Execution Unit (MDEU).”

14.1.3 Channel

The SEC includes one channel that manages data and EU function by using 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 the channel is idle and its fetch FIFO is non-empty, the channel reads the next descriptor pointer from the fetch FIFO. Using this pointer, the channel fetches the descriptor and places it in its descriptor buffer. To service this descriptor, the channel directs execution of the following steps.

1. Analyze the descriptor header to determine the cryptographic services required, and request use of the appropriate EUs from the controller.
2. Wait for the controller to grant access to the required EUs.
3. Set the appropriate mode bits in the EU(s) for the required service.
4. Fetch ‘data parcels’ using pointers from the descriptor buffer, and place them in either a 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 (initialization vector) 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.
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. Two values cam be written back: the first is identical to that of the header, with the exception that a ‘done’ field is set. The second is a status field writeback. That status field writeback occurs to report the result of ICV checking if ICV check writeback is enabled.

Many security protocols involve both encryption and hashing of packet payloads. To accomplish this without requiring two passes through the data, the channel 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’ SEC buses.

There are two types of snooping:

- Input data can be fed to the primary EU and the same input data snooped by the secondary EU. This is called ‘in-snooping.’
- Output data from the primary EU can be snooped by the secondary EU. This is called ‘out-snooping.’

In the SEC the secondary EU is always the MDEU.

For more information, refer to Section 14.5, “Channel.”

14.1.4 SEC Controller

The SEC controller manages on-chip resources, including the individual execution units (EUs), FIFOs, the master/slave interface to the MPC8323E system bus, and the internal buses that connect all the various modules. The controller receives service requests from the master/slave interface and from the channel, and schedules the required activities. The controller provides for two ways of operating the execution units:

- Channel-controlled access—The channel can request a particular service from any available execution unit. This is the normal operating condition.
- Host-controlled access—The host can move data into and out of any execution unit directly through memory-mapped EU registers. This is typically only used for debug.

The system bus interface and access to system memory are critical factors in performance, and the 64-bit master/slave interface of the SEC controller allows it to achieve performance unattainable on secondary buses.

14.1.4.1 Channel-Controlled Access

Processing begins when a descriptor pointer is written to the fetch FIFO of the channel. Based on the services requested by the descriptor header, the channel asks the controller to assign the necessary EUs to the channel. Once the required EU has been reserved, the channel requests that the controller fetch and load the appropriate data. The controller acts as a master on the system bus, reading and writing on byte

boundaries. The channel operates the EU, and makes further requests to the controller to write output data to system memory. When the descriptor processing is complete, the channel asks the controller to release the EU.

14.1.4.2 Host-Controlled Access

All execution units (EUs) are memory-mapped, and can be used entirely through register read/write access. The SEC operates as a slave, and the host must write the information typically provided through the descriptor into the appropriate registers and FIFOs of the SEC. This method is more CPU intensive, and requires a great deal of familiarity with SEC registers. It is recommended that host-controlled access be used only for operations using a single EU, and for debug purposes.

For more information, refer to Section 14.6, “Controller.”

14.2 Configuration of Internal Memory Space

Table 14-2 shows the base address map, while Table 14-3 provides the address map, including all registers in the execution units. The 18-bit SEC address bus value is shown. These address values are offsets from IMMRBAR. See Section 5.2.4.1, “Internal Memory Map Registers Base Address Register (IMMRBAR),” for more information.

Note that these tables show addresses of 64-bit words; the three least-significant address bits that are used to select bytes within 64-bit words are not shown.

Table 14-2. SEC Address Map

Address 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	Section 14.6, “Controller”
0x3_1100–0x3_11FF	Channel	Channel	Data control	Section 14.5, “Channel”
0x3_2000–0x3_2FFF	DEU	DES/3DES execution unit	EU	Section 14.4.1, “Data Encryption Standard Execution Unit (DEU)”
0x3_4000–0x3_4FFF	AESU	AES execution unit		Section 14.4.3, “Advanced Encryption Standard Execution Unit (AESU)”
0x3_6000–0x3_6FFF	MDEU	Message digest execution unit		Section 14.4.2, “Message Digest Execution Unit (MDEU)”

Table 14-3 shows the system address map showing all functional registers.

Table 14-3. SEC Address Map

Address Offset (AD 17–0)	Module	Register	Access	Access By	Section/Page
0x3_1008	Controller	IMR—Interrupt mask register	R/W	—	14.6.4.2/14-71
0x3_1010		ISR—Interrupt status register	R	—	14.6.4.3/14-73
0x3_1018		ICR—Interrupt clear register	W	—	14.6.4.4/14-74
0x3_1020		ID—Identification register	R	—	14.6.4.5/14-75
0x3_1028		EUASR—EU assignment status register	R	—	14.6.4.1/14-70
0x3_1030		MCR—Master control register	R/W	—	14.6.4.7/14-76
0x3_1108	Channel	CCCR—Crypto-channel configuration register	R/W	—	14.5.1.1/14-57
0x3_1110		CCPSR—Crypto-channel pointer status register	R	—	14.5.1.2/14-59
0x3_1140		CDPR—Crypto-channel current descriptor pointer register	R	—	14.5.1.3/14-64
0x3_1148		FF—Crypto-channel fetch FIFO register	W	—	14.5.1.4/14-64
0x3_1180–0x3_11BF		DB*n*—Descriptor buffers [0–7]	R	—	14.5.1.5/14-65
0x3_1BF8	Controller	IP block revision register	R	Word	14.6.4.6/14-76
0x3_2000	DEU	DEUMR—DEU mode register	R/W	—	14.4.1.1/14-20
0x3_2008		DEUKSR—DEU key size register (bytes)	R/W	—	14.4.1.2/14-21
0x3_2010		DEUDSR—DEU data size register (bits)	R/W	—	14.4.1.3/14-22
0x3_2018		DEURCR—DEU reset control register	R/W	Word	14.4.1.4/14-23
0x3_2028		DEUSR—DEU status register	R	Word	14.4.1.5/14-24
0x3_2030		DEUISR—DEU interrupt status register	R	Word	14.4.1.6/14-25
0x3_2038		DEUICR—DEU interrupt control register	R/W	Word	14.4.1.7/14-26
0x3_2050		DEUEMR—DEU end-of-message register	W	Word	14.4.1.8/14-28
0x3_2100		DEUIV—DEU IV register	R/W	—	14.4.1.9/14-28
0x3_2400		DEUK1—DEU key 1 register	W	—	14.4.1.10/14-29
0x3_2408		DEUK2—DEU key 2 register	W	—	14.4.1.10/14-29
0x3_2410		DEUK3—DEU key 3 register	W	—	14.4.1.10/14-29
0x3_2800–0x3_2FFF		DEU FIFO	R/W	—	14.4.1.11/14-29

Table 14-3. SEC Address Map (continued)

Address Offset (AD 17–0)	Module	Register	Access	Access By	Section/Page
0x3_4000	AESU	AESUMR—AESU mode register	R/W	—	14.4.3.1/14-41
0x3_4008		AESUKSR—AESU key size register (bytes)	R/W	—	14.4.3.2/14-44
0x3_4010		AESUDSR—AESU data size register (bits)	R/W	—	14.4.3.3/14-45
0x3_4018		AESURCR—AESU reset control register	R/W	Word	14.4.3.4/14-45
0x3_4028		AESUSR—AESU status register	R	Word	14.4.3.5/14-46
0x3_4030		AESUISR—AESU interrupt status register	R	Word	14.4.3.6/14-47
0x3_4038		AESUICR—AESU interrupt control register	R/W	Word	14.4.3.7/14-49
0x3_4050		AESUEMR—AESU end-of-message register	W	Word	14.4.3.8/14-50
0x3_4100–0x3_4108		AESU context memory registers	R/W	—	14.4.3.9/14-51
0x3_4400–0x3_4408		AESU key memory registers	R/W	—	14.4.3.9.5/14-55
0x3_4800–0x3_4FFF		AESU FIFO	R/W	—	14.4.3.9.6/14-55
0x3_6000	MDEU	MDEUMR—MDEU mode register	R/W	—	14.4.2.1/14-29
0x3_6008		MDEUKSR—MDEU key size register (bytes)	R/W	—	14.4.2.3/14-33
0x3_6010		MDEUDSR—MDEU data size register (bits)	R/W	—	14.4.2.4/14-33
0x3_6018		MDEURCR—MDEU reset control register	R/W	Word	14.4.2.5/14-34
0x3_6028		MDEUSR—MDEU status register	R	Word	14.4.2.6/14-35
0x3_6030		MDEUISR—MDEU interrupt status register	R	Word	14.4.2.7/14-36
0x3_6038		MDEUICR—MDEU interrupt control register	R/W	Word	14.4.2.8/14-37
0x3_6040		MDEU ICV size register	W	—	14.4.2.9/14-38
0x3_6050		MDEUEMR—MDEU end-of-message register	W	Word	14.4.2.10/14-39
0x3_6100–0x3_6120		MDEU context memory registers	R/W	—	14.4.2.11/14-39
0x3_6400–0x3_647F		MDEU key memory registers	W	—	14.4.2.12/14-40
0x3_6800–0x3_6FFF		MDEU FIFO	W	—	14.4.2.13/14-41

14.3 Descriptor Overview

The host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the host has determined that a security operation is required, it creates a ‘descriptor’ containing all the information the SEC needs to perform the security operation. The host creates the descriptor in main memory, then writes a pointer to the descriptor into the fetch FIFO of the

SEC channel. The channel uses this pointer to read the descriptor into its descriptor buffer. Once it obtains the descriptor, the SEC uses its bus mastering capability to obtain inputs and write results, thus off-loading data movement and encryption operations from the host processor.

For test purposes, it is also possible for the host to write keys, context, and text-data directly to execution units, using the SEC's host-controlled access. This method avoids use of descriptors.

14.3.1 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 14-3 for the descriptor format.

	015	16	17 23	24 31	3263
Header Dword	Header				Reserved
Pointer Dword 0	Length0	J0	Extent0	—	Pointer0
Pointer Dword 1	Length1	J1	Extent1	—	Pointer1
Pointer Dword 2	Length2	J2	Extent2	—	Pointer2
Pointer Dword 3	Length3	J3	Extent3	—	Pointer3
Pointer Dword 4	Length4	J4	Extent4	—	Pointer4
Pointer Dword 5	Length5	J5	Extent5	—	Pointer5
Pointer Dword 6	Length6	J6	Extent6	—	Pointer6

Figure 14-3. Descriptor Format

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.

14.3.2 Descriptor Format: Header Dword

Descriptors are created by the host to guide the SEC through required cryptographic operations. The header dword 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 fields that must be supplied to the SEC are shown in the ‘Field’ rows of Figure 14-4, and described in Table 14-4. The SEC device drivers allow the host to create proper headers for each cryptographic operation.

In processing a descriptor, the SEC can also write back certain fields to the header dword. These are shown in the ‘Writeback’ rows of Figure 14-4, and described in Table 14-5.

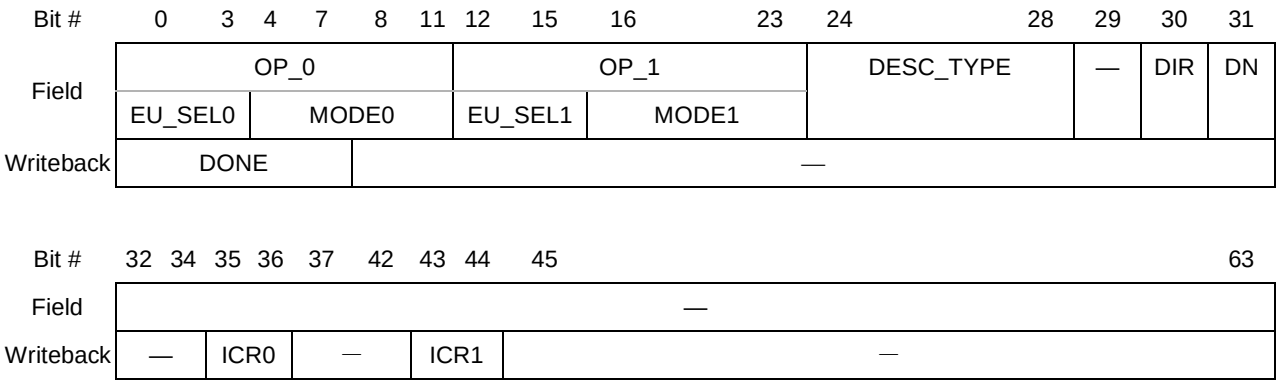


Figure 14-4. Header Dword

Table 14-4. Header Dword Bit Definitions

Bits	Name	Description
OP_0:		
0–3	EU_SEL0	Primary EU select. See Section 14.3.2.1, “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 14.3.2.1, “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, along with the DIR field, determines the sequence of actions to be performed by the channel and selected EUs using the blocks of data listed in the rest of the descriptor. The attributes determined include the direction of data flow for each data block, which EU (primary or secondary) is accessed, what snooping options are used, and which internal EU addresses are accessed. See Section 14.3.2.2, “Selecting Descriptor Type—DESC_TYPE,” for possible values.
29	—	Reserved.

Table 14-4. Header Dword Bit Definitions (continued)

Bits	Name	Description
30	DIR	Direction. Direction of overall data flow 0 Outbound 1 Inbound This, 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. This enables done notification if the NT field is 1 in the channel configuration register (see Table 14-31). The done notification can take the form of an interrupt, a writeback in the DONE field of this header dword (see Table 14-5), 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.

Table 14-5. Header Dword Writeback Bit Definitions

Bits	Name	Description
0–7	DONE	When done writeback is used, then at the completion of descriptor processing this byte is written with the value 0xFF. To determine when done writeback is used, see the CDWE, NT, and CDIE fields in the channel configuration register (Table 14-31).
8–34	—	Reserved.
35–36	ICR0	Integrity check result from primary. These bits are supplied by the primary EU when descriptor processing is complete. 00 No integrity check was performed. 01 The integrity check passed. 10 The integrity check failed. 11 Reserved
37–42	—	Reserved.
43–44	ICR1	Integrity check result from secondary. These bits are supplied by the secondary EU (if any) when descriptor processing is complete. 00 No integrity check was performed. 01 The integrity check passed. 10 The integrity check failed. 11 Reserved
45–63	—	Reserved.

14.3.2.1 Selecting Execution Units—EU_SEL0 and EU_SEL1

Table 14-6 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 'No EU selected' or 'Reserved' will result in an 'Unrecognized Header Error' condition during processing of the descriptor header.
2. The only valid choices for EU_SEL1 are 'No EU selected' or MDEU. Any other choice will result in an 'Unrecognized Header' error condition.

3. If EU_SEL1 is MDEU, then EU_SEL0 must be DEU or AESU. All other values of EU_SEL0 will result in an ‘Unrecognized header’ error condition.

Table 14-6. EU_SEL0 and EU_SEL1 Values

Value (Binary)	Selected EU
0000	No EU selected
0010	DEU
0011	MDEU
0110	AESU
Others	Reserved
1111	Reserved for header writeback

14.3.2.2 Selecting Descriptor Type—DESC_TYPE

Table 14-7 shows the permissible values for the DESC_TYPE field in the descriptor header. Descriptor types from SEC 1.0, which have ‘0’ in the last bit, are listed first, followed by SEC 2.x types, which have ‘1’ in the last bit.

Table 14-7. Descriptor Types

Value (Binary)	Descriptor Type	Notes
0000_0	aesu_ctr_nonsnoop	AESU CTR non-snooping[1]
0001_0	common_nonsnoop	Common, non-snooping
0010_0	hmac_snoop_no_afeu	Snooping, HMAC
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	—	Reserved
0110_ 0	—	Reserved
0111_0	—	Reserved
1000_0	—	Reserved
1001_0	—	Reserved
1010_0	—	Reserved
1011_0	—	Reserved
1100_0	hmac_snoop_aesu_ctr	AESU CTR hmac snooping[2]
1101_0	—	Reserved
1110_0	—	Reserved
1111_0	—	Reserved
0000_1	ipsec_esp	IPsec ESP mode encryption and hashing
0001_1	802.11i AES ccmp	CCMP encryption and hashing, suitable for IEEE 802.11i

Table 14-7. Descriptor Types (continued)

Value (Binary)	Descriptor Type	Notes
0010_1	srtp	SRTP encryption and hashing
0011_1	—	Reserved
0100_1	—	Reserved
0101_1	—	Reserved
0110_1	—	Reserved
0111_1	—	Reserved
1000_1	tls_ssl_block	TLS/SSL generic block cipher
1001_1	—	Reserved

Table 14-7. Descriptor Types (continued)

Value (Binary)	Descriptor Type	Notes
1010_1	raid_xor	XOR 3 sources together
Others	—	Reserved

Note:

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 [1]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 pre-pend zeros to the AES-Ctx before loading the AES context registers.

For more about descriptor types and the data used for each type, see Section 14.3.5, “Descriptor Types.”

14.3.3 Descriptor Format: Pointer Dwords

The descriptor contains seven ‘pointer dwords,’ which define where in memory the SEC should access its input and output data parcels. The pointer dwords are numbered 0–6 as shown in Figure 14-3. The channel determines how it will use each of the pointer dwords based on the ‘Descriptor Type’ and ‘Direction’ fields in the header. The channel accesses the first data parcel by starting at a location given by a POINTER value, and accessing a number of bytes given by a LENGTH or EXTENT value. Subsequent data parcels may be accessed by starting where a previous data parcel ended, or by starting at a different POINTER. The LENGTH or EXTENT used with any POINTER may be from the same pointer dword or from a different pointer dword in the same descriptor. Although the EXTENT field exists in each pointer dword of the SEC descriptor, only the EXTENTs in pointer dwords 3, 4, and 5 are currently in use.

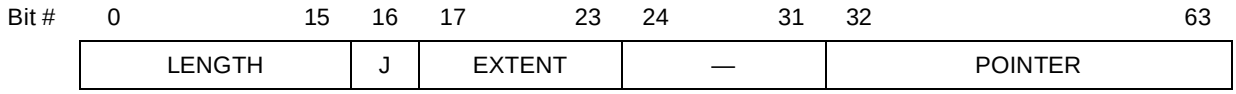


Figure 14-5. Pointer Dword

Table 14-8. Pointer Dword Field Definitions

Bits	Name	Description
0–15	LENGTH	Length. A number of bytes in the range 0–65535. 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	Extent. A number of bytes in the range 0–127. The use of this field depends on the ‘Descriptor Type’ and ‘Direction’ in the header dword.
24–31	—	Reserved
32–63	POINTER	Pointer: A memory address.

On occasion, a descriptor field may not be applicable to the requested service. With seven pointer dwords, it is possible that not all these dwords will be required to specify the input and output parameters. (Some operations, for example, do not require context.) Where a particular dword is not used, all fields should be set to 0.

Some descriptors involve more than seven parcels of input and output data. In these cases, it is necessary to use one POINTER field to address a sequence of data parcels.

LENGTH and EXTENT fields normally specify the sizes of data parcels. In some cases, however, the POINTER field is zero, and the LENGTH and/or EXTENT fields simply specify values to be written to an EU.

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 the 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 tables, and the J bit should be 1. For link table format, see Section 14.3.4, “Link Table Format.”

14.3.4 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.

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.

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, 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). A single link table entry is shown in Figure 14-6.

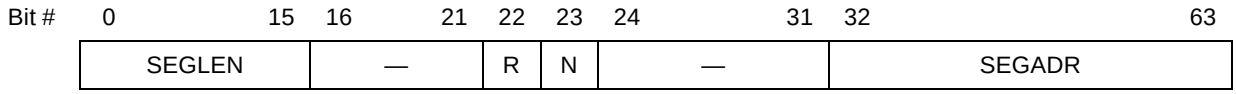


Figure 14-6. Link Table Entry

Table 14-9. Link Table Field Definitions

Bits	Name	Description
0–15	SEGLEN	Length. When N=0, a number in the range 1–65535, specifying the number of bytes in the memory segment. pointed to by SEGADR. A value of 0 will cause an error state to be set in the channel pointer status register—G-STATE for a gather operation or S-STATE for a scatter operation (see Section 14.5.1.2, “Crypto-Channel Pointer Status Register (CCPSR)”). When N = 1, must be 0.
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 G-STATE or S-STATE error will be set in the channel pointer status register (see Section 14.5.1.2, “Crypto-Channel Pointer Status Register (CCPSR)”). When N = 1, 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 state in the channel pointer status register—G-STATE for gather error or S-STATE for scatter error (see Section 14.5.1.2, “Crypto-Channel Pointer Status Register (CCPSR)”).

Example (from Figure 14-6): To demonstrate use of a link table, assume that the current descriptor type calls for the channel to read 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 ‘gather table buffer.’

Using the first entry of the gather table buffer, 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 gather table buffer, 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 gather table buffer is exhausted, the channel reads the next four long words of the link table into its gather table buffer. If a gather table buffer entry is encountered in which the N bit is set, the channel uses the SEGADR field in that word to find the next link table in the chain. The last byte of the required parcel size (Extent3) must coincide with the last byte of a memory segment, or unpredictable results may occur.

Now assume that the channel accesses its next data parcel using Pointer3 again, this time with length given by Length3. In this case the channel continues to the next line of the link table, and begins reading the memory segment specified there. As before, the channel concatenates memory segments from as many link table entries as necessary to obtain the required number of bytes (Length3).

Similarly, the next data parcel is obtained by using Pointer3 yet again, this time with length given by Extent4.

Assume that for the current descriptor type, the Extent4 data parcel is the last one to be accessed through Pointer3. Then the link table entry that supplies the last memory segment for Extent4 has the R bit set, signifying that this is the last entry in the chain of link tables.

14.3.5 Descriptor Types

An example of 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 is shown below.

Table 14-10. Descriptor Format by Type

Descriptor Type	field type	Pointer Dword1	Pointer Dword2	Pointer Dword 3	Pointer Dword4	Pointer Dword 5	Pointer Dword 6	Pointer Dword 7
0000_0 aesu_ctr_ nosnoop	Length	nil	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out	nil
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0001_0 common_ nosnoop	Length	nil	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out	nil
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
1100_0 hmac_snoop_ aesu_ctr	Length	HMAC Key	HMAC Data	AES Key	AES Ctx	In FIFO	Out FIFO	HMAC Out
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0000_1 ipsec_esp	Length	HMAC Key	HMAC Data	Cipher IV	Cipher Key	In FIFO	Out FIFO	Cipher IV Out
	Extent	undefined	undefined	undefined	nil	HMAC In	HMAC Out	undefined
0001_1 ccmp	Length	nil	AES-Ctx	AES Key	In FIFO	In FIFO	Out FIFO	AES-Ctx-Out
	Extent	undefined	undefined	undefined	nil	nil	nil	undefined
0010_1 srtp	Length	HMAC Key	AES-Ctx	AES Key	In FIFO	Out FIFO	HMAC Out	AES-Ctx-Out
	Extent	undefined	undefined	undefined	In FIFO	In FIFO	nil	undefined
1000_1 outbound tls_ssl_ block	Length	MAC Key	Cipher IV	Cipher Key	In FIFO Auth & CIpher	In FIFO Cipher Only	Out FIFO	Cipher IV Out
	Extent	undefined	undefined	undefined	In FIFO Auth only	MAC Out	nil	undefined

Table 14-10. Descriptor Format by Type (continued)

Descriptor Type	field type	Pointer Dword1	Pointer Dword2	Pointer Dword 3	Pointer Dword4	Pointer Dword 5	Pointer Dword 6	Pointer Dword 7
1000_1 inbound	Length	MAC Key	Cipher IV	Cipher Key	nil	In FIFO Auth & CIpher	Out FIFO	Cipher IV Out
tls_ssl_ block	Extent	undefined	undefined	undefined	In FIFO Auth only	MAC In	MAC Out	undefined
1010_1	Length	nil	nil	nil	In1 (opt)	In2	In3	Out
raid_xor	Extent	nil	nil	nil	undefined	undefined	undefined	undefined
others		Reserved						

14.4 Execution Units

Execution unit (EU) is the term used for a functional block that performs the mathematical manipulations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, SSL/TLS, iSCSI, SRTP, and 802.11i processing, and can work together to perform high level cryptographic tasks.

The following execution units are used in the SEC 2.2:

- Data encryption standard execution unit (DEU) for DES and 3DES as specified by FIPS 46-3
- Advanced encryption standard execution unit (AESU) implementing the Rinjdael symmetric key cipher per FIPS-197. AESU can perform AES modes CBC, ECB, and CTR modes per NIST SP 800-38A, and CCM mode per NIST SP 800-38C.
- Message digest execution unit (MDEU), implementing MD5 per RFC 1321, and SHA-1, SHA-224, and SHA-256 per FIPS-180-2. In addition, HMAC is implemented per FIPS 198.

In addition, the two symmetric execution units, DEU and AESU, share their input and output FIFOs.

Working together, the EUs can perform high-level cryptographic tasks, such as IPSec Encapsulating Security Protocol (ESP). The remainder of this chapter provides details about the execution units themselves.

14.4.1 Data Encryption Standard Execution Unit (DEU)

This section contains details about the data encryption standard execution unit (DEU), including modes of operation, status and control registers, and shared symmetric FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the DEU is used through channel-controlled access, which means that most reads and writes of DEU registers are directed by the SEC channel. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

14.4.1.1 DEU Mode Register (DEUMR)

The DEU mode register (DEUMR) contains three bits that are used to program DEU operation.

The 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.

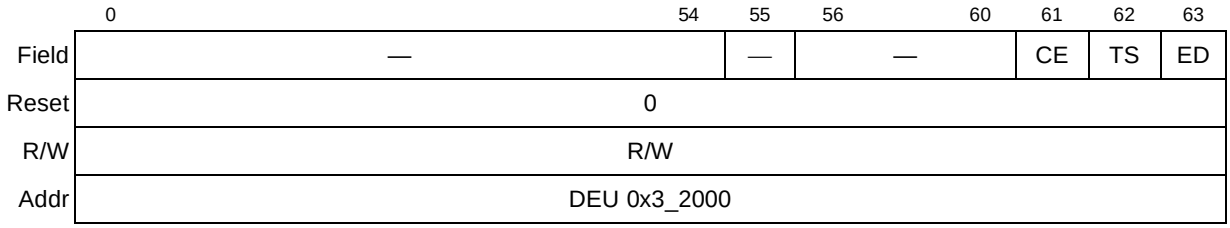


Figure 14-7. DEU Mode Register (DEUMR)

Table 14-11 describes DEUMR fields.

Table 14-11. DEUMR Field Descriptions

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–54	—	Reserved
55	—	Reserved
The following bits are controlled through the MODE0 field of the descriptor header.		
56–60	—	Reserved
61	CE	CBC/ECB. If set, the 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, the DEU operates the Triple DES algorithm; if not set, DEU operates the single DES algorithm. 0 Single DES 1 Triple DES
63	ED	Encrypt/decrypt. If set, the DEU operates the encryption algorithm; if not set, DEU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption

14.4.1.2 DEU Key Size Register (DEUKSR)

The value of the DEU key size register (DEUKSR), shown in Figure 14-8, indicates the number of bytes of key memory that should be used in encrypting or decrypting. If the DEUMR is set for single DES, any value other than 8 bytes will automatically generate a key size error in the DEU interrupt status register (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 (any write to K1 duplicates that value into K3 in case 2-key 3DES is desired).

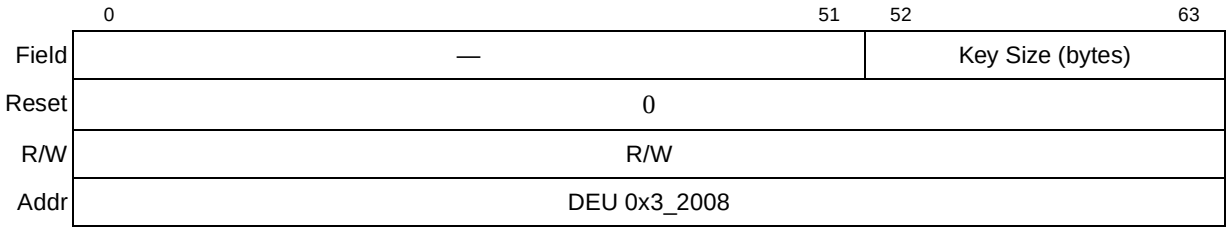


Figure 14-8. DEU Key Size Register (DEUKSR)

Table 14-12 shows the DEUKSR fields.

Table 14-12. DEUKSR Field Descriptions

Bits	Name	Description
0–51	—	Reserved
52–63	Key Size	8 bytes = 0x08 (only legal value if mode is single DES) 16 bytes = 0x10 (for 2 key 3DES, K1 = K3) 24 bytes = 0x18 (for 3 key 3DES)

14.4.1.3 DEU Data Size Register (DEUDSR)

The DEU data size register (DEUDSR), shown in Figure 14-9, stores the number of bits in the final message block, which must be 64. All data to be processed by the DEU must be a multiple of the DES algorithm block size of 64 bits; the DEU does not automatically pad messages out to 64-bit blocks. If a data size that is not a multiple of 64 bits is written, a data size error will be generated. Only bits 58–63 are checked to determine if there is a data size error. Because all upper bits are ignored, the entire message length (in bits) can be written to this register.

DEUDSR is cleared when the DEU is reset or re-initialized.

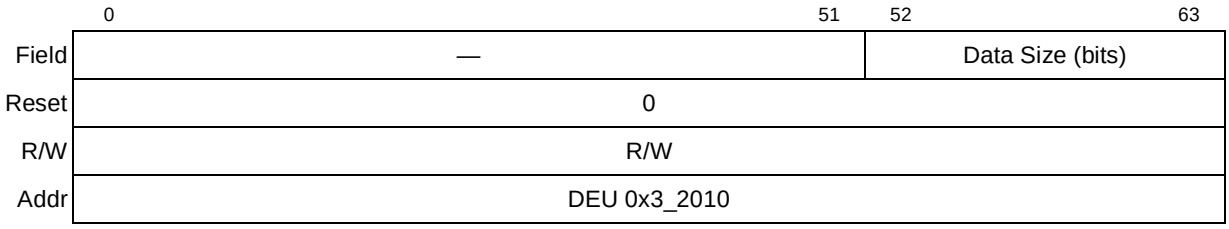


Figure 14-9. DEU Data Size Register (DEUDSR)

14.4.1.4 DEU Reset Control Register (DEURCR)

The DEU reset control register (DEURCR), shown in Figure 14-10, allows three levels reset of just DEU, as defined by the three self-clearing bits.

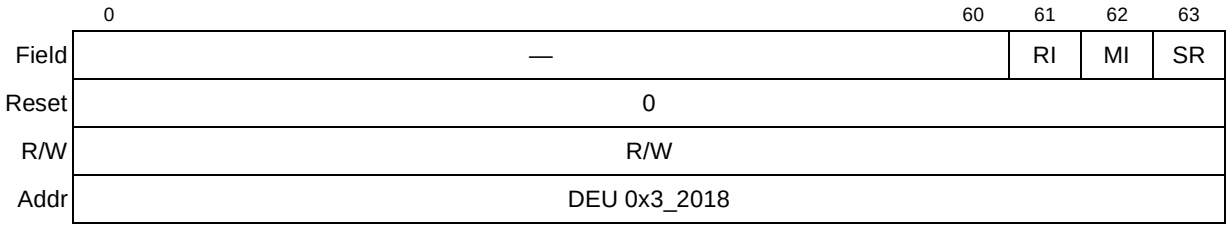


Figure 14-10. DEU Reset Control Register (DEURCR)

Table 14-13 describes DEURCR fields.

Table 14-13. 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 the DEU interrupt status register (DEUISR). 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. this module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the DEU status register 0 Do not 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 the DEU status register (DEUSR) will indicate when this initialization routine is complete 0 Do not reset 1 Full DEU reset

14.4.1.5 DEU Status Register (DEUSR)

The DEU status register (DEUSR), displayed in Figure 14-11, contains 6 fields that reflect the state of DEU internal signals. The DEUSR is read-only. Writing to this location will result in address error being reflected in the DEU interrupt status register (DEUISR).

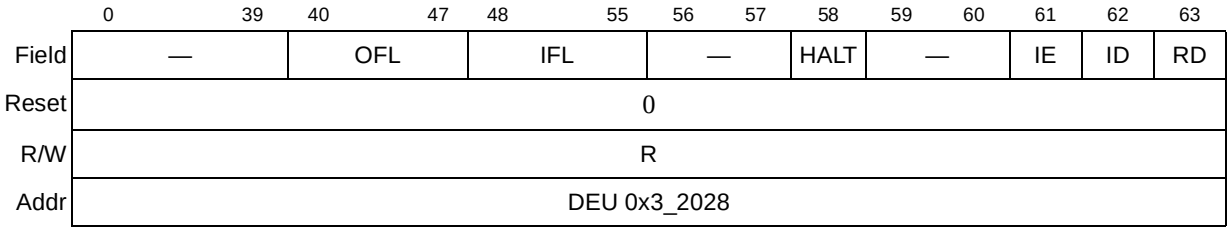


Figure 14-11. DEU Status Register (DEUSR)

Table 14-14 describes the DEUSR fields.

Table 14-14. 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 before reaching the interrupt status register (ISR), the DEU interrupt status register (DEUISR) 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 14.6.4.3, “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 14.6.4.3, “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 reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

14.4.1.6 DEU Interrupt Status Register (DEUISR)

The DEU interrupt status register (DEUISR), shown in Figure 14-12, records occurrences of errors. Each bit in the DEUISR can only be set if the corresponding bit of the DEU interrupt control register (DEUICR) is zero (see Section 14.4.1.7, “DEU Interrupt Control Register (DEUICR)”).

If the DEUISR is non-zero, the DEU halts and the DEU error interrupt signal is asserted to the controller (see Section 14.6.4.3, “Interrupt Status Register (ISR)”). In addition, if the DEU is being operated through channel-controlled access, an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the channel pointer status register (see Table 14-35) and generates a channel error interrupt to the controller.

	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	R														
Addr	DEU 0x3_2030														

Figure 14-12. DEU Interrupt Status Register (DEUISR)

Table 14-15 describes DEUISR fields.

Table 14-15. DEUISR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. Defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are checked for parity only if the appropriate DEU mode register bit indicates triple DES. Also, key register 3 is checked only if key size reg = 24. Key register 2 is checked only if key size reg = 16 or 24.) 0 No error detected 1 Key parity error
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error **Note:** 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
53	CE	Context error. A DEU key register, the key size register, data size register, mode register, or IV register was modified while DEU was performing encryption. 0 No error detected 1 Context error
54	KSE	Key size error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for triple DES) was written to the DEU key size register 0 No error detected 1 Key size error

Table 14-15. DEUISR Field Descriptions (continued)

Bits	Name	Description
55	DSE	Data size error (DSE): A value was written to the DEU data size register that is not a multiple of 64 bits. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
60	IFU	Input FIFO underflow. The DEU input FIFO has been read while empty. 0 No error detected 1 Input FIFO has had underflow error
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 operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO underflow. The DEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
63	OFO	Output FIFO overflow. The DEU output FIFO has been pushed while full. 0 No error detected 1 Output FIFO has overflowed

14.4.1.7 DEU Interrupt Control Register (DEUICR)

The DEU interrupt control register (DEUICR), shown in Figure 14-13, controls the result of detected errors. For a given error (as defined in Section 14.4.1.6, “DEU Interrupt Status Register (DEUISR)”), if the corresponding bit in the DEUICR is set, the error is ignored; no bit is set in the DEUISR, and no error interrupt occurs. If the corresponding bit is not set, then upon detection of an error, the 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	3000														
R/W	R/W														
Addr	DEU 0x3_2038														

Figure 14-13. DEU Interrupt Control Register (DEUICR)

Table 14-16 describes DEUICR fields.

Table 14-16. DEUICR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. The defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are only checked for parity if the appropriate DEU mode register bit indicates triple DES. 0 Key parity 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. A DEU key register, the key size register, the data size register, the mode register, or IV register was modified while DEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for Triple DES) was written to the DEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error (DSE): A value that is not a multiple of 64 bits was written to the DEU data size register. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The shared symmetric 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

Table 14-16. DEUICR Field Descriptions (continued)

Bits	Name	Description
59	IFE	Input FIFO error. The shared symmetric input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	IFU	Input FIFO underflow. The shared symmetric input FIFO has been read while empty. 0 Input FIFO underflow error enabled 1 Input FIFO underflow error disabled
61	IFO	Input FIFO overflow. The shared symmetric input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled **Note:** When operated through channel-controlled access, SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO underflow. The shared symmetric 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 shared symmetric output FIFO has been pushed while full. 0 Output FIFO overflow error enabled 1 Output FIFO overflow error disabled

14.4.1.8 DEU End-of-Message Register (DEUEMR)

The DEU end-of-message register (DEUEMR), shown in Figure 14-14, indicates a DES operation may be completed. After the final message block is written to the input FIFO, the DEUEMR 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 the DEUEMR is merely a trigger causing the DEU to process the final block of a message, allowing it to signal DONE.

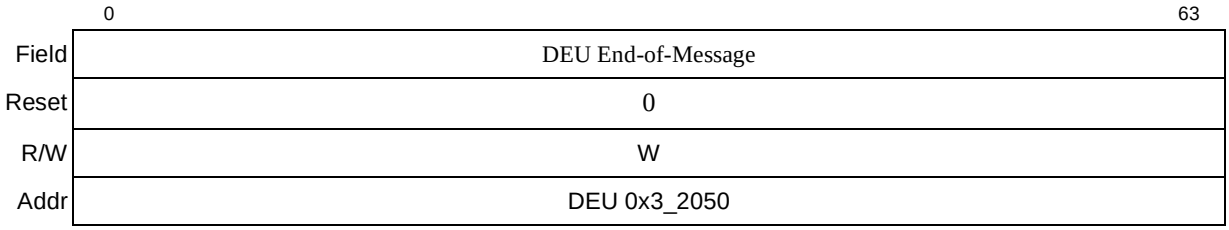


Figure 14-14. DEU End-of-Message Register (DEUEMR)

14.4.1.9 DEU IV Register (DEUIV)

For CBC mode, the initialization vector is written to and read from the DEU IV register (DEUIV). The value of this register changes as a result of the encryption process and reflects the context of the DEU. Reading this memory location while the module is processing data generates an error interrupt.

14.4.1.10 DEU Key Registers (DEUK*n*)

The DEU uses three write-only key registers (DEUK1, DEUK2, and DEUK3) 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 the key size register 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 a write to DEUK2 and DEUK3.

Reading any of these memory locations generates an address error interrupt.

14.4.1.11 DEU FIFOs

DEU uses the symmetric shared 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 shared symmetric input FIFO, and for that FIFO to be configured as reserved for DEU, and a read from anywhere in the DEU FIFO address space causes a 64-bit-word to be popped off of the shared symmetric output FIFO. Overflows and underflows caused by reading or writing the shared symmetric FIFOs are reflected in the DEUISR

14.4.2 Message Digest Execution Unit (MDEU)

This section contains details about the message digest execution unit (MDEU), including modes of operation, status and control registers, and FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the MDEU is used through channel-controlled access, which means that most reads and writes of MDEU registers are directed by the SEC channel. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

14.4.2.1 MDEU Mode Register (MDEUMR)

The MDEU mode register (MDEUMR) is used to program the function of the MDEU. Bits 56–63 of the MDEUMR are specified by the user through the MODE0 or MODE1 field of the descriptor header. The remaining bits are supplied by the channel and thus are not under direct user control.

The MDEUMR has two configurations, determined by the value of the NEW bit (see Figure 14-15 and Figure 14-16). The ‘old’ configuration (NEW = 0) is used by most descriptor types and is backward compatible with previous versions of SEC 2.0. The ‘new’ configuration (NEW = 1) is only used by descriptor type 1000_1 ‘tls_ssl_block’.

The MDEUMR is cleared when the MDEU 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 is generated.

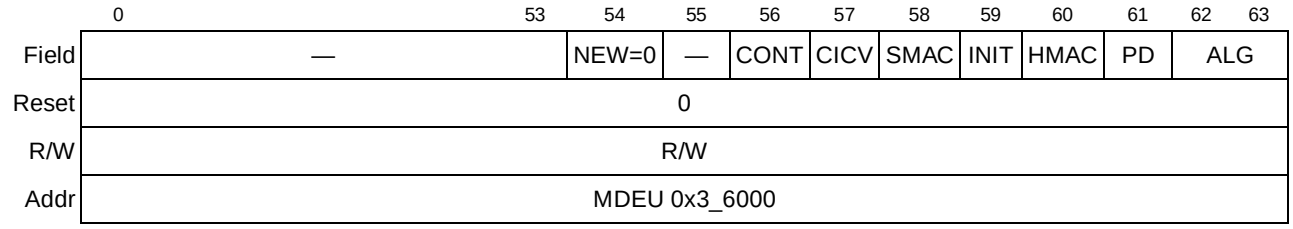


Figure 14-15. MDEU Mode Register (MDEUMR) in 'Old' Configuration

Table 14-17 describes MDEUMR fields in 'old' configuration.

Table 14-17. MDEUMR in 'Old' Configuration

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–53	—	Reserved
54	NEW=0	Determines the configuration of the MDEU mode register (MDEUMR). This table shows the configuration for NEW = 0.
55	—	Reserved, must be set to zero
The following bits are controlled through the MODE0 or MODE1 fields of the descriptor header.		
56	CONT	Continue. Most operations will require this bit to be cleared. It is set only when the data to be hashed is spread across multiple descriptors. The value programmed in PD must be opposite to the value in this bit. 0 Do autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash will be continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare integrity check values 0 Normal operation; no ICV comparison 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV size register. Only applicable to descriptor types that provide for reading an ICV in value.
58	SMAC	Specifies whether to perform an SSL-MAC operation 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization bit. Most operations will require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit must be 0 on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.

Table 14-17. MDEUMR in ‘Old’ Configuration (continued)

Bits	Name	Description
61	PD	This bit must be programmed opposite to the CONT bit.
62–63	ALG	Message digest algorithm selection 00 SHA-160 algorithm (full name for SHA-1) 01 SHA-256 algorithm 10 MD5 algorithm 11 SHA-224 algorithm

	0 — 52	53	54	55	56	57	58	59	60	61	62 63
Field	—	STIB	NEW=1	—	CONT	CICV	SMAC	INIT	HMAC	EALG	ALG
Reset	0										
R/W	R/W										
Addr	MDEU 0x3_6000										

Figure 14-16. MDEU Mode Register (MDEUMR) in ‘New’ Configuration

Table 14-18 describes MDEUMR fields in ‘new’ configuration.

Table 14-18. MDEUMR in ‘New’ Configuration

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–52	—	Reserved
53	STIB	SSL/TLS inbound, block cipher 0 Normal operation. 1 Special operation only for SSL/TLS inbound, block cipher. Upon receiving end-of-message, the MDEU performs a calculation involving the last valid byte of data written into its input FIFO (which is Pad Length) to compute a final data size. The MDEU then processes the amount of data specified by this data size, and completes the message digest.
54	NEW=1	Determines the configuration of the MDEU mode register (MDEUMR). This table shows the configuration for NEW = 1.
55	—	Reserved, must be set to zero
The following bits are controlled through the MODE0 or MODE1 fields of the descriptor header.		
56	CONT	Continue. Most operations will require this bit to be cleared. Set only when the data to be hashed is spread across multiple descriptors. 0 Do autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash will be continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare integrity check values 0 Normal operation; no ICV comparison. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU’s input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV size register.

Table 14-18. MDEUMR in 'New' Configuration (continued)

Bits	Name	Description
58	SMAC	Specifies whether to perform an SSL-MAC operation 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization bit. Most operations will require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit is set on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	EALG	The EALG (Extended Algorithm bit) and ALG (Algorithm) bits together specify the message digest algorithm, as follows: 000 SHA-160 algorithm (full name for SHA-1) 001 SHA-256 algorithm 010 MD5 algorithm 011 SHA-224 algorithm Others: Reserved
62–63	ALG	

14.4.2.2 Recommended Settings for MDEUMR

The most common task likely to be executed through the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPSec, and TLS. The SSL 3.0 protocol uses a slightly different 'SSL-MAC'. If an HMAC or SSL-MAC is to be performed using a single descriptor (with the MDEU acting as sole or secondary EU), the following mode register bit settings should be used:

Table 14-19. Mode Register—HMAC or SSL-MAC Generated by Single Descriptor

Bits	Field	Value	
		For HMAC	For SSL-MAC
56	CONT	0 (off)	0 (off)
58	SMAC	0 (on)	1(on)
59	INIT	1(on)	1(on)
60	HMAC	1 (on)	0 (on)

To generate an HMAC for a message that is spread across a sequence of descriptors, the following mode register bit settings should be used:

Table 14-20. Mode Register—HMAC Generated Across a Sequence of Descriptors

Bits	Field	Value		
		First Descriptor	Middle Descriptor(s)	Final Descriptor
56	CONT	1 (on)	1 (on)	0 (off)
59	INIT	1 (on)	0 (off)	0 (off)
60	HMAC	1 (on)	0 (off)	1 (on)

All descriptors other than the final descriptor must output the intermediate message digest for the next descriptor to reload as MDEU context.

SSL-MAC operations cannot be spread across a sequence of descriptors.

Additional information on descriptors can be found in Section 14.3, “Descriptor Overview.”

14.4.2.3 MDEU Key Size Register (MDEUKSR)

The MDEU key size register (MDEUKSR), shown in Figure 14-17, 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 the MDEUKSR exceeds 64 bytes.

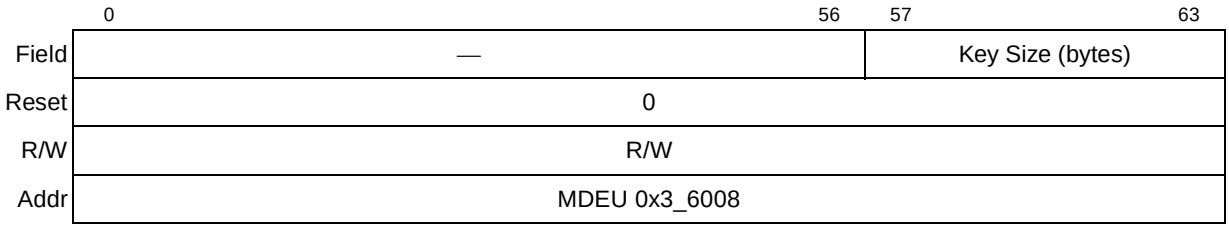


Figure 14-17. MDEU Key Size Register (MDEUKSR)

14.4.2.4 MDEU Data Size Register (MDEUDSR)

The MDEU data size register (MDEUDSR), shown in Figure 14-18, indicates the number of bits of data to be processed. The MDEU decrements this number as it processes data. A read of this register provides a snapshot of how much data remains to be processed.

The Data Size field is a 21-bit signed number. Values written to this register are added to the current register value. Multiple writes are allowed. The MDEU processes data when there is a positive value in this register and there is data available in the private MDEU input FIFO. (Negative values can arise in inbound processing, when it is necessary to hold back data from the MDEU until the pad length has been decrypted.)

Because the MDEU does not support bit offsets, bits 61–63 must be written as 0. Furthermore, when the CONT bit of the MDEU mode register (MDEUMR) is high, the data size must be a multiple of the 512-bit block size (that is, bits 55–63 must be written as 0). Violating either of these conditions causes a data size error (DSE in the MDEUISR).

The MDEUDSR is cleared when the MDEU is reset or re-initialized. At the end of processing, its contents has been decremented down to zero (unless there is an error interrupt).

NOTE

Writing to the MDEUDSR allows the MDEU to enter auto-start mode. Therefore, the required context registers must be written prior to writing the data size.

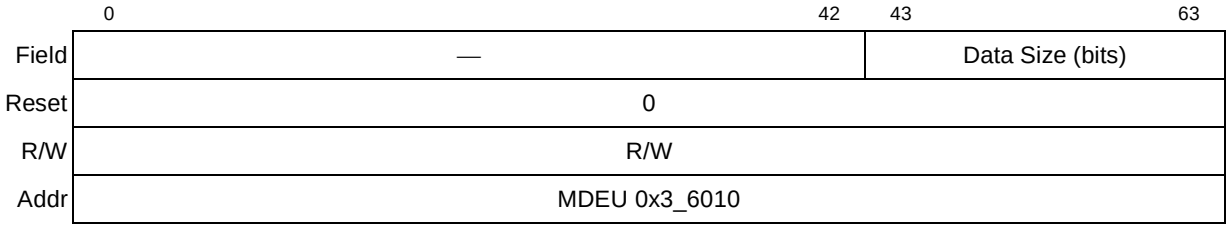


Figure 14-18. MDEU Data Size Register (MDEUDSR)

14.4.2.5 MDEU Reset Control Register (MDEURCR)

The MDEU reset control register (MDEURCR), shown in Figure 14-19, allows three levels reset of just the MDEU, as defined by the three self-clearing bits.

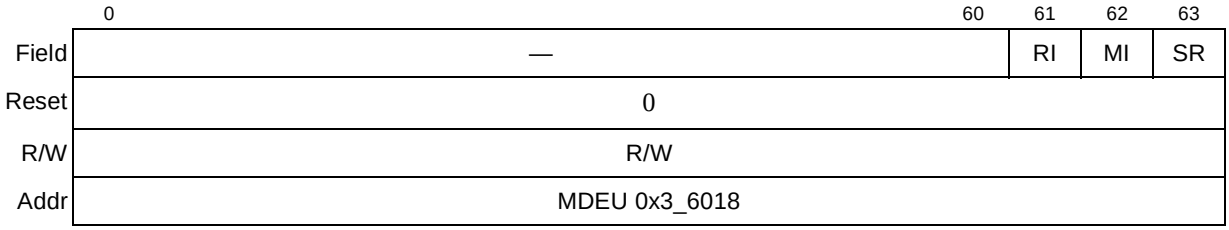


Figure 14-19. MDEU Reset Control Register (MDEURCR)

Table 14-21 describes MDEURCR fields.

Table 14-21. 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 RESET# pin), but only for the MDEU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full MDEU reset

14.4.2.6 MDEU Status Register (MDEUSR)

The MDEU status register (MDEUSR), as seen in Figure 14-20, reflects the state of the MDEU internal signals. The majority of these internal signals reflect the state of low-level MDEU functions, such as data padding, key padding, and so on, 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.

The MDEUSR is read-only. Writing to this location will result in address error being reflected in the MDEUISR.

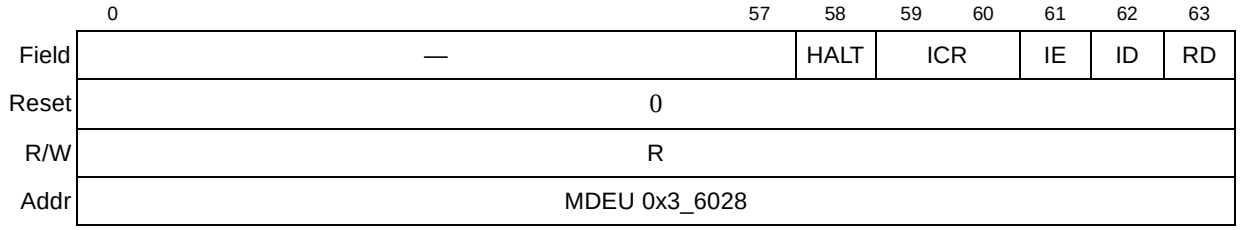


Figure 14-20. MDEU Status Register (MDEUSR)

Table 14-22 describes the MDEUSR fields.

Table 14-22. 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 before reaching the interrupt status register, the MDEUISR is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICR	Integrity check result 00 No integrity check was performed. 01 The integrity check passed. 10 The integrity check failed. 11 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 14.6.4.3, “Interrupt Status Register (ISR)”). 0 MDEU is not signaling error 1 MDEU is signaling error

Table 14-22. MDEUSR Field Descriptions (continued)

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 14.6.4.3, “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 reset sequence, as reflected in the signal sampled by the channel. 0 Reset in progress 1 Reset done **Note:** Reset done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

14.4.2.7 MDEU Interrupt Status Register (MDEUISR)

The MDEU interrupt status register (MDEUISR), shown in Figure 14-21, tracks the state of possible errors, if those errors are not masked, through the MDEU interrupt control register (MDEUICR).

	0	49	50	51	52	53	54	55	56	57	58 60	61	62 63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—
Reset	64’h1000												
R/W	R												
Addr	MDEU 0x3_6030												

Figure 14-21. MDEU Interrupt Status Register (MDEUISR)

Table 14-23 describes the MDEUISR fields.

Table 14-23. MDEUISR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error 0 No error detected 1 Integrity check error detected. An ICV check was performed and the supplied ICV did not match the one computed by the MDEU.
50	—	Reserved
51	IE	Internal error. Indicates the MDEU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected **Note:** This bit 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

Table 14-23. MDEUISR Field Descriptions (continued)

Bits	Name	Description
53	CE	Context error. The MDEU key register, key size register, or data size register 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 the MDEU key size register (MDEUKSR). 0 No error detected 1 Key size error
55	DSE	Data size error. A value not a multiple of 512 bits while the MDEU mode register (MDEUMR) CONT bit is high. 0 No error detected 1 Data size error
56	ME	Mode error. Will be set if any of these error conditions is detected: • Any reserved bit of the mode register is set • The ALG field of the mode register contains an illegal value 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 operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input size. When operated through host-controlled access, the MDEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62–63	—	Reserved

14.4.2.8 MDEU Interrupt Control Register (MDEUICR)

The MDEU interrupt control register (MDEUICR), shown in Figure 14-22, controls the result of detected errors. For a given error (as defined in Section 14.4.2.7, “MDEU Interrupt Status Register (MDEUISR)”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the MDEU interrupt status register (MDEUISR) is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

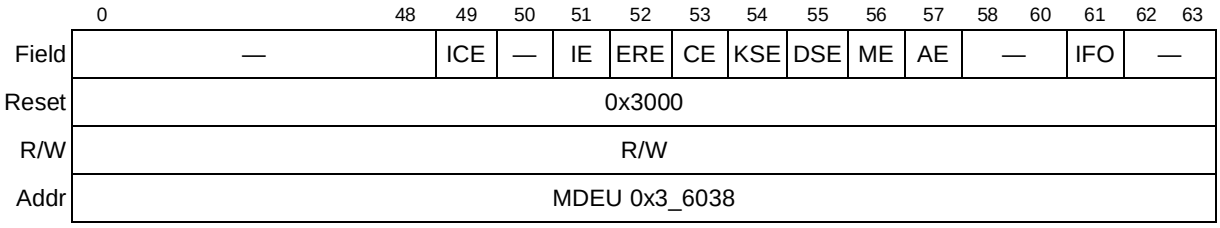


Figure 14-22. MDEU Interrupt Control Register (MDEUICR)

Table 14-23 describes the MDEUICR fields.

Table 14-24. MDEUICR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the MDEU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
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, the key size register, the data size register, or the mode register, 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 64 bytes was written to the MDEU key size register (MDEUKSR). 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value was written to the MDEU data size register (MDEUDSR). 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register (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

14.4.2.9 MDEU ICV Size Register

The MDEU ICV size register, shown in Figure 14-23, stores the number of bytes of the ICV result to be compared if the MDEU performs ICV comparison (see Section 14.4.2.1 "MDEU Mode Register (MDEUMR)").

The MDEU ICV size register is cleared when the MDEU is reset or re-initialized.

	0 — 56	57 — 63
Field	—	ICV_Size
Reset	0	
R/W	R/W	
Addr	MDEU 0x3_6040	

Figure 14-23. MDEU ICV Size Register

14.4.2.10 MDEU End-of-Message Register (MDEUEMR)

The end-of-message register in the MDEU (MDEUEMR), shown in Figure 14-24, is used to indicate that an authentication operation may be completed. After the final message block is written to the private MDEU input FIFO, the MDEUEMR must be written. The value in the data size register will be used to determine how many bits of the final message block (always 512) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to the MDEUEMR is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal DONE.

	0 — 63
Field	MDEU End-of-Message
Reset	0
R/W	W
Addr	MDEU 0x3_6050

Figure 14-24. MDEU End-of-Message Register (MDEUEMR)

14.4.2.11 MDEU Context Registers

In the MDEU, context consists of the hash plus the message length count. Write access to the MDEU context register block allows continuation of a previous hash. Reading these registers provide the resulting message digest or HMAC, along with an aggregate bit count.

NOTE

SHA-1, SHA-224, and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU mode register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location while the module is not done will generate an error interrupt.

After a power-on reset, all the MDEU context register values are cleared to 0. Figure 14-25 shows how the MDEU context registers are initialized if the INIT bit is set in the MDEU mode register. All registers are

initialized, regardless of mode selected, however only the appropriate context register values are used in hash generation per the mode selected. The user typically does not care about the MDEU context register initialization values, however they are documented for completeness in the event the user reads these registers using host-controlled access. MDEU reset through the MDEU reset control register (MDEURCR) (Figure 14-19) or SEC global software reset (Figure 14-47) 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-224	0xC1059ED8	0x367CD507	
SHA-256	0x6A09E667	0xBB67AE85	
Name	**C**	**D**	Context offset 0x3_6108
MD-5	0xFEDCBA98	0x76543210	
SHA-1	0x98BADCFE	0x10325476	
	0x3070DD17	0xF70E5939	
SHA-256	0x3C6EF372	0xA54FF53A	
Name	**E**	**F**	Context offset 0x3_6110
MD-5	0xF0E1D2C3	0x8C68059B	
SHA-1	0xC3D2E1F0	0x9B05688C	
	0xFFC00B31	0x68581511	
SHA-256	0x510E527F	0x9B05688C	
Name	**G**	**H**	Context offset 0x3_6118
MD-5	0xABD9831F	0x19CDE05B	
SHA-1	0x1F83D9AB	0x5BE0CD19	
	0x64F98FA7	0xBEFA4FA4	
SHA-256	0x1F83D9AB	0x5BE0CD19	
Name	**Message Length Count**		Context offset 0x3_6120
Reset	0		

Figure 14-25. MDEU Context Registers

14.4.2.12 MDEU Key Registers

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required.

NOTE

SHA-1, SHA-224, and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register (MDEUMR) indicates that MD5 is the hash of choice.

14.4.2.13 MDEU FIFOs

MDEU uses a private 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, SHA-224, and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register (MDEUMR) indicates that MD5 is the hash of choice.

14.4.3 Advanced Encryption Standard Execution Unit (AESU)

This section contains details about the advanced encryption standard execution unit (AESU), including modes of operation, status and control registers, and FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the AESU is used through channel-controlled access, which means that most reads and writes of AESU registers are directed by the SEC channel. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

14.4.3.1 AESU Mode Register (AESUMR)

The AESU mode register (AESUMR), shown in Figure 14-26, contains 7 bits that are used to program the AESU.

AESUMR is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

	0–50	51–53	54–55	56–57	58	59	60	61–62	63
Field	—	SCM	—	ECM	FM	IM	RDK	CM	ED
Reset	0								
R/W	R/W								
Addr	AESU 0x3_4000								

Figure 14-26. AESU Mode Register (AESUMR)

Table 14-25 describes AESUMR fields.

Table 14-25. AESUMR Field Descriptions

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–50	—	Reserved
51–53	SCM	Sub-cipher-mode. Specifies additional options specific to particular cipher modes. • XOR cipher mode: specifies the number of sources to be XORed together. Valid values are 2 and 3. • For all other cipher modes, this field must be 0.
54–55	—	Reserved, must be set to zero.
The following bits are controlled through the MODE0 field of the descriptor header.		
56–57	ECM	Extend cipher mode. Used in combination with bits 61–62 “Cipher Mode” to define the mode of AES operation. See Table 14-26 for mode bit combinations.
58	FM	Final MAC (FM). 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(IM). Initializes AESU for new message (CCM mode only) 0 Do not initialize (context will be loaded by host) 1 Initialize new message with nonce
60	RDK	Restore decrypt key (RDK). Specifies that key data write will contain pre-expanded key (decrypt mode only). See note below 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 bits 56–57 (Extend Cipher Mode) to define the mode of AES operation. See Table 14-26 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).

Table 14-26. AES Cipher Modes

Mode	ECM (56–57)	CM (61–62)
ECB	00	00
CBC	00	01
CTR	00	11
SRT[1]	01	11
CCM (without ICV comparison)	10	00
CCM with ICV comparison	11	00

Table 14-26. AES Cipher Modes (continued)

Mode	ECM (56–57)	CM (61–62)
XOR	11	11
Reserved	all others	

Note:

1 SRT is not a new AES mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0 ‘srtp’. See Section 14.4.3.9.3, “Context for SRT Mode,” for more information on how SRT mode reduces context loading overhead.

NOTE

Note on Restore decrypt key (RDK)—In most networking applications, the decryption of an AES protected packet will be performed as a single operation. However, if circumstances dictate that the decryption of a message should be split across multiple descriptors, the AESU allows the user to save the decrypt key, and the active AES context, to memory for later re-use. This saves the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data to be decrypted.)

The use of RDK is completely optional, as the Input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

To use RDK, the following procedure is recommended:

The descriptor type used in decryption of the first portion of the message is ‘0100_0- AESU Key Expand Output’. The AESU mode must be ‘Decrypt’. See Table 14-7 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).

14.4.3.2 AESU Key Size Register (AESUKSR)

The AESU key size register (AESUKSR), shown in Figure 14-27, 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. The AESUKSR 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 the key size register is modified during processing, a context error will be generated.

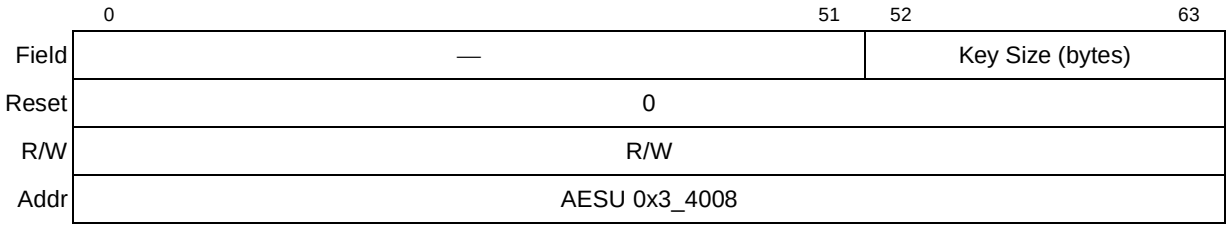


Figure 14-27. AESU Key Size Register (AESUKSR)

14.4.3.3 AESU Data Size Register (AESUDSR)

The AESU data size register (AESUDSR), shown in Figure 14-28, stores the number of bits in the final message block. Acceptable sizes vary depending on the AES mode selected. In ECB, CBC, and CTR mode, the message processed by the AESU must be a multiple of 128 bits; the AESU does not automatically pad messages out to 128-bit blocks. In CCM mode, data size must be a multiple of 8 bits. In XOR mode the data size must be a multiple of 256 bits (32 bytes). If an improper data size is written, a data size error is generated. Only the lowest 3, 7, or 8 bits of the data size register are checked to determine if there is a data size error. Because all upper bits are ignored, the entire message length (in bits) can be written to this register.

The AESUDSR is cleared when the AESU is reset or re-initialized.

Writing to the AESUDSR signals the AESU to start processing data from the shared symmetric input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

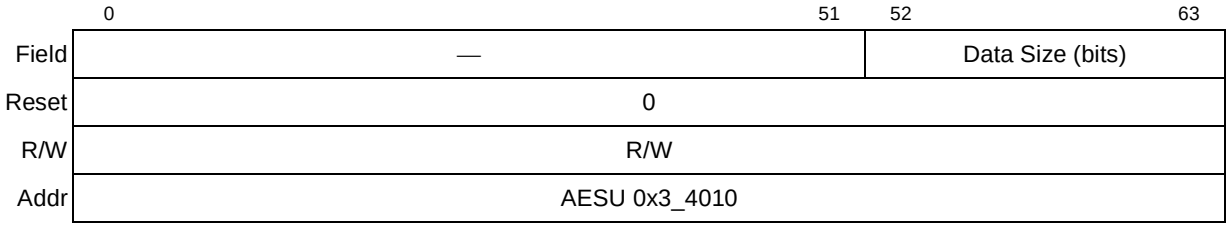


Figure 14-28. AESU Data Size Register (AESUDSR)

14.4.3.4 AESU Reset Control Register (AESURCR)

The AESU reset control register (AESURCR), shown in Figure 14-29, allows three levels reset of just AESU, as defined by the three self-clearing bits.

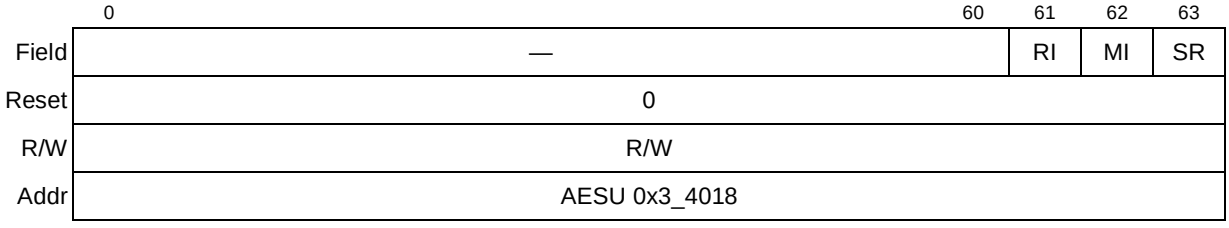


Figure 14-29. AESU Reset Control Register (AESURCR)

Table 14-13 describes AESURCR fields.

Table 14-27. 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 (AESUISR). 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. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the AESU status register (AESUSR) 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 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

14.4.3.5 AESU Status Register (AESUSR)

The AESU status register (AESUSR), shown in Figure 14-30, is a read-only register that reflects the state of six status outputs. Writing to this location will result in an address error being reflected in the AESU interrupt status register (AESUISR).

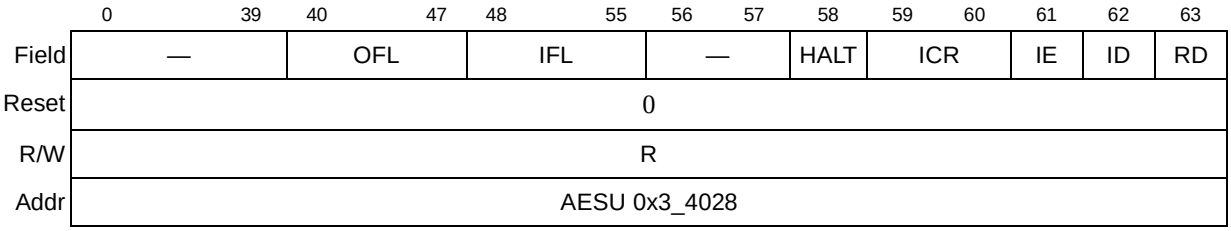


Figure 14-30. AESU Status Register (AESUSR)

Table 14-28 describes AESUSR fields.

Table 14-28. 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 14-28. AESUSR Field Descriptions (continued)

Bits	Name	Description
58	HALT	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 before reaching the interrupt status register, the AESU interrupt status register (AESUISR) is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICR	Integrity check result 00 No integrity check was performed. 01 The integrity check passed. 10 The integrity check failed. 11 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 14.6.4.3, “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 14.6.4.3, “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 reset sequence, as reflected in the signal sampled by the channel. 0 Reset in progress 1 Reset done **Note:** Reset done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

14.4.3.6 AESU Interrupt Status Register (AESUISR)

The AESU interrupt status register (AESUISR), shown in Figure 14-31, tracks the state of possible errors, if those errors are not masked, through the AESU interrupt control register (AESUICR).

	0–48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	—
Reset	0															
R/W	R															
Addr	AESU 0x3_4030															

Figure 14-31. AESU Interrupt Status Register (AESUISR)

Table 14-29 describes AESUISR fields.

Table 14-29. AESUISR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error 0 No error detected 1 Integrity check error detected. An ICV check was performed and the supplied ICV did not match the one computed by the MDEU.
50	—	Reserved
51	IE	Internal error. 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 (AESUICR) or by resetting the AESU.
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 (AESUKSR). 0 No error detected 1 Key size error
55	DSE	Data size error (DSE): A value was written to the AESU data size register that is not a proper size. See Section 14.4.3.3, “AESU Data Size Register (AESUDSR).” 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 shared symmetric 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 shared symmetric 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 AESU input FIFO has been read while empty. 0 No error detected 1 Input FIFO has had underflow error

Table 14-29. AESUISR Field Descriptions (continued)

Bits	Name	Description
61	IFO	Input FIFO overflow. The shared symmetric Input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the AESU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO underflow. The shared symmetric output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

14.4.3.7 AESU Interrupt Control Register (AESUICR)

The AESU interrupt control register (AESUICR), shown in Figure 14-32, controls the result of detected errors. For a given error (as defined in Section 14.4.3.6, "AESU Interrupt Status Register (AESUISR)"), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the interrupt status register (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.

	0 — 48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	RSV	IFO	OFU	—
Reset	1000															
R/W	R/W															
Addr	AESU 0x3_4038															

Figure 14-32. AESU Interrupt Control Register (AESUICR)

Table 14-30 describes the AESUICR fields.

Table 14-30. AESUICR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the MDEU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved
51	IE	Internal error. An internal processing error 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

Table 14-30. AESUICR Field Descriptions (continued)

Bits	Name	Description
53	CE	Context error. An AESU key register, the key size register, data size register, mode register, or IV register was modified while the AESU was processing. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. An inappropriate value (non 16, 24 or 32 bytes) was written to the AESU key size register (AESUKSR) 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 shared symmetric output FIFO was detected non-empty upon write of AESU data size register (AESUDSR) 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The shared symmetric input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The shared symmetric 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 shared symmetric output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.4.3.8 AESU End-of-Message Register (AESUEMR)

The AESU end-of-message register (AESUEMR), shown in Figure 14-33, is used to indicate an AES operation may be completed. After the final message block is written to the shared symmetric input FIFO, the AESUEMR must be written. The value in the data size register (AESUDSR) is used to determine how many bits of the final message block (always 128) will be processed. Writing to the AESUEMR causes the AESU to process the final block of a message, allowing it to signal DONE. A read of the AESUEMR will always return a zero value.

	0	63
Field	AESU End of Message	
Reset	0	
R/W	W	
Addr	AESU 0x3_4050	

Figure 14-33. AESU End-of-Message Register (AESUEMR)

14.4.3.9 AESU Context Registers

There are seven 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 Exponent when in CTR mode. The contents of the ‘empty’ context registers need not be preserved, but when restoring the CTR mode context, the ‘empty’ registers must be filled with 32 bytes of zeros before writing the saved Counter and Counter Modulus Exponent.

Context should be loaded with the lower bytes in the lowest 64-bit context register. The context registers are summarized in Figure 14-34.

Context Register (64 bits each)

Cipher Mode	1	2	3	4	5	6	7
ECB	—	—	—	—	—	—	—
CBC	IV1[1]	IV2[1]	—	—	—	—	—
CTR	—	—	—	—	Counter [1]		Counter Modulus Exponent[1]
SRT	Counter[1]		Counter Modulus Exponent (M)[1]	—	—	—	—
CCM	IV[1] / MAC Tag		Encrypted MAC[2]/Decrypted MAC/Encrypted Counter		Counter[1]		Counter Modulus Exponent[1]/header size/MAC size[3]

1 Must be written at the start of a new message.

2 Must be written at start of new CCM decryption.

3 Header size/MAC size is only used if AES-CCM processing is suspended and resumed.

Figure 14-34. AESU Context Registers

14.4.3.9.1 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 14.4.3.5, "AESU Status Register (AESUSR)." If the IV registers are read prior to assertion of Interrupt Done, an early read error will be generated.

The IV registers must be read when changing context and restored to resume processing an interrupted message (CBC mode only).

14.4.3.9.2 Context for Counter Mode

In counter mode, a random 128-bit initial counter value is incremented modulo 2^M with each block processed. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent M can be set between 8 and 128 in multiples of 8. The value of M is specified by writing to context register 3 as described in Figure 14-34.

14.4.3.9.3 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^M with each block processed. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent M can be set between 8 and 128 in multiples of 8. The value of M is specified by writing to context register 3 as described in Figure 14-34.

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.

14.4.3.9.4 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. The context register contents for CCM mode is summarized in Figure 14-35 and further described below.

		Context Registers						
		1	2	3	4	5	6	7
Encrypt (outbound)	Inputs	IV		0		Initial Counter		Counter Modulus Exponent
	Outputs	MAC	0	MIC	0	—		—q
Decrypt (inbound)	Inputs	IV		MIC	0	Initial Counter		Counter Modulus Exponent
	Outputs	Computed MAC	0	Decrypted MAC	0	—		—

Figure 14-35. AESU CCM Context Registers

The context for CCM encryption/MAC generation is:

- Reg 1–2, session specific 128-bit initialization vector (from memory)
- Reg 3–4, 128 bits of zero padding
- Reg 5–6, session specific counter (initial counter value) (from memory)
- Reg 7, counter modulus exponent (msb<--lsb). Should be fixed at 0x0000_0080.

Note that the counter modulus for CCM mode is currently defined as 2^{128} making the exponent 128. This value has been made programmable in the SEC in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU context register prior to CCM encryption.

CCM encryption processing—With the session specific key and context, the AESU will perform the following operations.

1. Initialize the IV, and encrypt with the symmetric key.
2. In CBC fashion, take the output of step 1, hash with the first block of plaintext, and encrypt with the symmetric key.
3. Continue as in step 2 until the final block of plaintext has been processed. The result of the encryption of the final block of plaintext with the symmetric key is the MAC tag. The full 128 bits of MAC data is written to context registers 1–2, for use in the next phase of CCM processing.

Once the MAC tag has been generated (step 3), the MAC tag, along with the plaintext is encrypted with the AESU operating in counter mode.

4. The first item to be encrypted in counter mode is the counter (initial counter value) from context registers 5–6. The counter is encrypted with the symmetric key, and the result is hashed with the MAC tag (retrieved from context registers 1–2) to produce the MIC (encrypted MAC), which is then stored in context registers 3–4. At the completion of CCM encrypt processing, this MIC is output to memory (per the descriptor pointer) for the host to append to the 802.11i frame. Note that the MIC 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 MIC.

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 shared symmetric 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 MIC have been output, the CCM encrypt operation is complete.

The context for CCM decryption/MAC generation is:

- Reg 1–2, session specific 128-bit initialization vector (from memory)
- Reg 3–4, MIC (from received frame) + 64 bits of zero padding
- Reg 5–6, session specific counter (initial counter value) (from memory)
- Reg 7, counter modulus exponent (msb<--lsb). Should be fixed at 0x0000_0080.

Note that the counter modulus for CCM mode is currently defined as 2^{128}, making the exponent 128. This value has been made programmable in the SEC in case the final version of IEEE 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU context register prior to CCM decryption.

CCM decryption processing is the reverse of encryption;

With the session specific key and context, the AESU will perform the following operations.

1. Initialize the IV, and encrypt with the symmetric key. Simultaneously, the counter (initial counter value) from context registers 5–6 is encrypted with the symmetric key. The result is hashed with the encrypted MAC (from context registers 3–4), and the resulting original MAC is written to context registers 3–4, overwriting the encrypted MAC.

   Note that the counter is encrypted with the symmetric key, however, the AESU should be set for ‘decrypt’ to perform the counter and CBC processes in the correct order.
2. The IEEE 802.11 frame header is hashed with the encrypted IV. (The AESU automatically determines the header length.) Simultaneously, the counter is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of ciphertext to produce the first block of plaintext. The plaintext is placed in the shared symmetric output FIFO, while simultaneously, in CBC fashion, a copy of the first block of plaintext is hashed with the output of encryption of the IEEE 802.11 frame header. The output is encrypted with the symmetric key.
3. As each ciphertext block is converted to plaintext, the plaintext is CBC encrypted. When the final plaintext block has been processed, the CBC MAC (MAC tag) is written to context registers 1–2. The first 64 bits of the MAC tag are compared to the MAC tag recovered in step 1.

Note that for both encrypt and decrypt operations, if the IEEE 802.11 frame is being processed as a whole (not split across multiple descriptors), the ‘Initialize’ and ‘Final MAC’ bits should be set in the AESU mode register.

14.4.3.9.5 AESU Key Registers

The AESU key registers hold from 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to key 1. Any key data written to bytes beyond the value written to the key size register will be ignored. The key data registers are cleared when the AESU is reset or re-initialized. If these registers are modified during message processing, a context error will be generated.

The key data registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the ‘restore decrypt key’ bit must be set in the mode register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

14.4.3.9.6 AESU FIFOs

The AESU fetches data 128 bits at a time from the shared symmetric 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 shared symmetric output FIFO. The output size is the same as the input size.

Writing to the AESU FIFO address space places 64 bits of message data into the input FIFO and configures the shared symmetric FIFOs to be reserved by AESU. 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 AESU FIFO address space will pop 64 bits of message data from the shared symmetric output FIFO. The output FIFO may be read any time the OFR signal is asserted (as indicated in the AESU status register). This will indicate that the number of bytes in the output FIFO is at or above the threshold specified in the mode register.

14.5 Channel

The channel in the SEC manages the execution of each cryptographic task, making use of one or more of the SEC’s execution units (EUs). Control information and data pointers for a given task are stored in the form of a descriptor (see Section 14.3.1, “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, and writes a pointer to the descriptor into the channel’s fetch FIFO. The fetch FIFO can store up to 24 pointers. Operations performed by the channel 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 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. For the RAID-XOR descriptor type, the channel rotates among three data sources, fetching 32 bytes from each source.
- 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 end-of-message 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 end-of-message.
- 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 an interrupt.

The channel will wait indefinitely for the controller to complete a requested activity before continuing to the next step of descriptor processing.

The channel can generate two types of done notification signals when it completes operation on a descriptor—an interrupt and/or a writeback of the descriptor header. The done interrupt is enabled by the CDIE bit and the done writeback is enabled by the CDWE bit of the channel configuration register (Table 14-31). Table 14-5 shows the DONE field that is written back in if writeback is enabled. In addition, status writeback can also be used to signal processing completion. Any descriptor can have status writeback occur as a result of the AWSE bit of the channel configuration register. Or, by setting IWSE, status writeback will occur when any ICV-checking descriptor completes.

The selected done notification can be performed at the end of processing of every descriptor, or only on selected descriptors. If the NT field is 0 in the channel configuration register, then done notification is performed after every descriptor. If the NT field is 1, done notification is only performed on descriptors in which the DN bit is set in the packet header (Table 14-4).

14.5.1 Channel Registers

14.5.1.1 Crypto-Channel Configuration Register (CCCR)

The crypto-channel configuration register (CCCR) contains five operational bits permitting configuration of the channel as shown in Figure 14-36. Table 14-31 describes the CCR.

	0	29	30	31
Field	—		CON	R
Reset	0			
R/W	R/W			
Addr	Channel_1 0x3_1108			

	32	54	55	56	57	58	59	60	61	62	63
Field	—		BS	IWSE	AWSE	EAE	CDWE	—	NT	CDIE	—
Reset	0										
R/W	R/W										
Addr	Channel_1 0x3_110C										

Figure 14-36. Crypto-Channel Configuration Register (CCCR)

Table 14-31. CCCR Field Descriptions

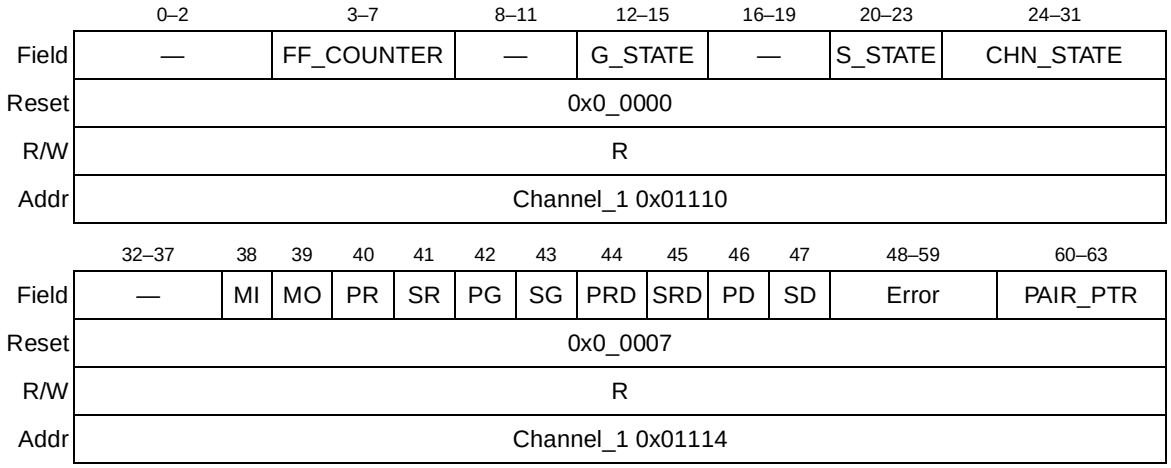
Bits	Names	Description
0–29	—	Reserved, set to zero
30	CON	Continue bit 0 No special action. 1 Causes the same channel reset actions as bit R, except that the fetch FIFO and the lower half of the CCR register are not cleared. After the reset sequence is complete, this bit automatically returns to 0 and the channel resumes normal operation, servicing the next descriptor pointer in the fetch FIFO, if any.
31	R	Reset channel 0 No special action. 1 Causes a software reset of the channel, clearing all its internal state. The details of the software reset actions depend upon what the channel is doing when the bit is set: • If the R bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request by asserting the release output signals. The channel then resets all its registers, clears the R bit, and return the channel state machine to the idle state. • If the R bit is set after the channel has been assigned an EU, the channel requests a write from the controller to set the software reset bit of the EU. If a secondary EU has been reserved, the channel requests a write to reset that EU as well. The channel next asserts the appropriate release signal to notify the controller that the channel has finished with the reserved EU(s). The channel then resets all the registers, clears the RESET bit and returns the channel state machine to the idle state.
32–54	—	Reserved, set to zero
55	BS	Burst size—The SEC accesses long text-data parcels in main memory through bursts of programmable size: 0 Burst size is 64 bytes 1 Burst size is 128 bytes

Table 14-31. CCCR Field Descriptions (continued)

Bits	Names	Description
56	IWSE	ICV writeback status enable 0 No special action. 1 If the descriptor calls for ICV comparison, then at the completion of descriptor processing, write back the status of all EUs into the header dword.
57	AWSE	Always writeback status enable 0 No special action. 1 At the completion of processing each descriptor, write back the status of all EUs into the header dword. In this case, IWSE has no effect.
59	CDWE	Channel done writeback enable 0 Channel done writeback disabled. 1 Channel done writeback enabled. Upon completion of descriptor processing, if the NT bit is set for Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, notify the host by writing back the descriptor header with the writeback information shown in Table 14-5. This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed.
60	—	Reserved, set to zero
61	NT	Notification type. This bit controls when the channel will generate channel done notification. Channel done notification can take the form of an interrupt or modified header writeback or both, depending on the state of the CDIE and CDWE control bits. 0 Global notification. The channel will generate channel done notification (if enabled) at the end of each descriptor. 1 Selected notification. The channel will generate channel done notification (if enabled) at the end of every descriptor with the DONE bit set in the descriptor header.
62	CDIE	Channel done interrupt enable 0 Channel done interrupt disabled 1 Channel Done Interrupt enabled. Upon completion of descriptor processing, if the NT bit is set for Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, then notify the host by asserting an interrupt. Refer to Section 14.5.2, “Channel Interrupts,” for complete description of channel interrupt operation.
63	—	Reserved, set to zero

14.5.1.2 Crypto-Channel Pointer Status Register (CCPSR)

The crypto-channel pointer status register (CCPSR) contains status fields and counters that 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	0x0_0000						
R/W	R						
Addr	Channel_1 0x01110						

	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	0x0_0007												
R/W	R												
Addr	Channel_1 0x01114												

Figure 14-37. Crypto-Channel Pointer Status Register (CCPSR)

Table 14-32 describes the CCPSR fields.

Table 14-32. CCPSR Field Descriptions

Bits	Names	Description
0–2	—	Reserved
3–7	FF_COUNTER	Fetch FIFO counter. 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. This field reflects the state of the channel Gather control state machine. The value of this field indicates which stage the channel is while performing gather function. Table 14-33 shows the meaning of all possible values of the G_STATE field. 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. This field reflects the state of the channel Scatter control state machine. The value of this field indicates which stage the channel is while performing scatter function. Table 14-33 shows the meaning of all possible values of the S_STATE field. S_State is documented for information only. The User will not typically care about the scatter state machine.
24–31	CHN_STATE	State. State of the channel state machine. This field reflects the state of the channel control state machine. The value of this field indicates exactly which stage the channel is in the sequence of fetching and processing data descriptors. Table 14-34 shows the meaning of all possible values of the STATE field. **Note:** CHN_State is documented for information only. The User will not typically care about the channel state machine.
32–37	—	Reserved, set to zero

Table 14-32. CCPSR Field Descriptions (continued)

Bits	Names	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 channel is requesting assignment of primary EU to the channel. The channel will assert the EU request signal indicated by the op0 field in the Descriptor Header register as long as this bit remains set. The PRI_REQ bit is set when descriptor processing is initiated by the channel and the Op_0 field in the descriptor header contains a valid EU identifier. 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 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 by the channel 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. The PRI_GRANT bit reflects the state of the EU grant signal for the requested primary EU from the controller. 0 The primary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested primary EU to the channel.
43	SG	Secondary EU granted. The SEC_GRANT bit reflects the state of the EU grant signal for the requested secondary EU from the controller. 0 The secondary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested secondary EU to the channel.
44	PRD	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
45	SRD	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.

Table 14-32. CCPSR Field Descriptions (continued)

Bits	Names	Description
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.
47	SD	Secondary EU done. The SEC_DONE bit reflects the state of the done interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
48–59	Error	Channel error status. This field reflects the error status of the 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 channel or writing the appropriate registers to initiate the processing of a new descriptor. Table 14-35 lists the conditions which can cause a channel error and how they are represented in the ERROR field.
60–63	PAIR_PTR	Descriptor buffer register length/pointer pair. This field indicates which of the length/pointer pairs are currently being processed by the channel. Table 14-36 shows the meaning of all possible values of the PAIR_PTR field.

Table 14-33 shows the values for G_STATE and S_STATE fields, which are the states for the gather and scatter state machines).

Table 14-33. G_STATE and S_STATE Field Values

Value	Gather State Machine:
0x0	GS_IDLE
0x1	GS_LOAD_POINTER
0x2	GS_LOAD_POINTER_DONE
0x3	GS_LOAD_NEXT_POINTER
0x4	GS_PROCESS_POINTER
0x5	GS_TRANS_BLOCK
0x6	GS_TRANS_BLOCK_DONE
0x7	GS_TRANS_BYTES
0x8	GS_TRANS_BYTES_DONE
0x9	GS_INC_PAIR_PTR
0xA	GS_UPDATE
0xB	GS_DONE
0xC	GS_ERROR
0xD	GS_RELOAD
0xE	GS_TRANS_INBOUND
0xF	GS_TRANS_INBOUND_DONE

Table 14-34 shows the values of channel states.

Table 14-34. CHN_STATE Field Values

Value	Channel State	Value	Channel State
0x00	IDLE	0x22	EVALUATE_RESET
0x01	PROCESS_HEADER	0x23	RESET_WRITE_RESET_PRI
0x02	FETCH_DESCRIPTOR	0x24	RESET_RELEASE_PRI_CHA
0x03	CHANNEL_DONE	0x25	RESET_WRITE_RESET_SEC
0x04	CHANNEL_DONE_IRQ	0x26	RESET_RELEASE_SEC_CHA
0x05	CHANNEL_DONE_WRITEBACK	0x27	RESET_CHANNEL
0x06	CHANNEL_DONE_NOTIFICATION	0x28	WRITE_DATASIZE_PRI_POST
0x07	CHANNEL_ERROR	0x29	RESET_RELEASE_ALL
0x08	REQUEST_PRI_CHA	0x2A	RESET_RELEASE_ALL_DELAY
0x09	INC_DATA_PAIR_POINTER	0x2B	REQUEST_SEC_CHA
0x0A	DELAY_DATA_PAIR_UPDATE	0x2C	WRITE_DATASIZE_SEC
0x0B	EVALUATE_DATA_PAIRS	0x2D	WRITE_ICV_SIZE
0x0C	WRITE_RESET_PRI	0x2E	WRITE_SEC_CHA_GO_SNOOPOUT
0x0D	RELEASE_PRI_CHA	0x2F	WRITE_PRI_CHA_GO_SNOOPIN
0x0E	WRITE_RESET_SEC	0x30	WRITE_SEC_CHA_GO_SNOOPIN
0x0F	RELEASE_SEC_CHA	0x31	DELAY_1CYCLE
0x10	PROCESS_DATA_PAIRS	0x33	TRANS_EXTENT_READ
0x11	WRITE_MODE_PRI	0x34	TRANS_EXTENT3
0x12	WRITE_MODE_SEC	0x35	TRANS_EXTENT4
0x13	WRITE_DATASIZE_PRI	0x36	XOR_WRITE_READ_REG
0x14	DELAY_RNGA_DONE	0x37	DELAY_SEC_DONE_TLS
0x15	WRITE_DATASIZE_SEC_SNOOPIN	0x38	MAC_TO_CIPHER
0x16	TRANS_REQUEST_WRITE_SNOOPIN	0x39	MAC_TO_CIPHER_DONE
0x17	DELAY_PRI_SEC_DONE	0x3A	READ_PRI_STATUS
0x18	TRANS_REQUEST_WRITE	0x3C	READ_SEC_STATUS
0x19	WRITE_KEY_SIZE	Others	Reserved
0x1B	DELAY_PRI_DONE		
0x1E	WRITE_DATASIZE_SEC_SNOOPOUT		
0x1F	TRANS_REQUEST_READ_SNOOPOUT		
0x20	DELAY_SEC_DONE		
0x21	TRANS_REQUEST_READ		

Table 14-35 shows the bit positions of each potential error. Multiple errors are possible.

Table 14-35. Crypto-Channel Pointer Status Register Error Field Definitions

Value	Error
48	DOF. Double fetch FIFO write overflow error. This bit is set when the channel Fetch FIFO is full, SOF is set, and another write has been made to the fetch FIFO. When this bit is set the 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 CCR register. This bit can be cleared by writing '1' to this bit in the CPSR 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 channel will set this bit and activate an error interrupt. The channel continues processing, but the descriptor pointer is lost. The host must clear this bit by writing '1' to this bit in the CPSR 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 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 op0 field in descriptor header.[1] • Invalid secondary EU indicated by op1 field in descriptor header. • Descriptor type field in descriptor header indicates secondary EU transaction when not in snoop mode
54	Invalid EU assignment request. Indicates the channel has been assigned one or more EUs not requested by the descriptor header.
55	EU error detected. An EU assigned to this channel has generated an error interrupt. This error may also be reflected in the controller's interrupt status register.
56	Gather boundary error. Indicates a gather pointer straddles both a primary and secondary EU's data transfer.
57	Gather return/length error. Indicates the total data size covered by a gather link table did not match the total data size from the main descriptor.
58	Scatter boundary error. Indicates a scatter pointer straddles both a primary and secondary EU's data transfer.
59	Scatter return/length error. Indicates the total data size covered by a scatter link table did not match the total data size from the main descriptor.

1 Invalid opcode includes any opcode valid on other versions of the SEC but not valid on SEC 2.2.

NOTE

EU error bit (bit 55) can only be cleared by first clearing the error source in the assigned EU which caused it to be set.

Table 14-36 shows the possible values of the PAIR_PTR field in the CCPSR.

Table 14-36. Crypto-Channel Pointer Status Register PAIR_PTR Field Values

Value	Error
0x00	Processing header or pointer dword 0
0x01	Processing pointer dword 1
0x02	Processing pointer dword 2

Table 14-36. Crypto-Channel Pointer Status Register PAIR_PTR Field Values (continued)

Value	Error
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

14.5.1.3 Crypto-Channel Current Descriptor Pointer Register (CDPR)

The crypto-channel current descriptor pointer register (CDPR), shown in Figure 14-38, contains the address of the descriptor which the channel is currently processing.

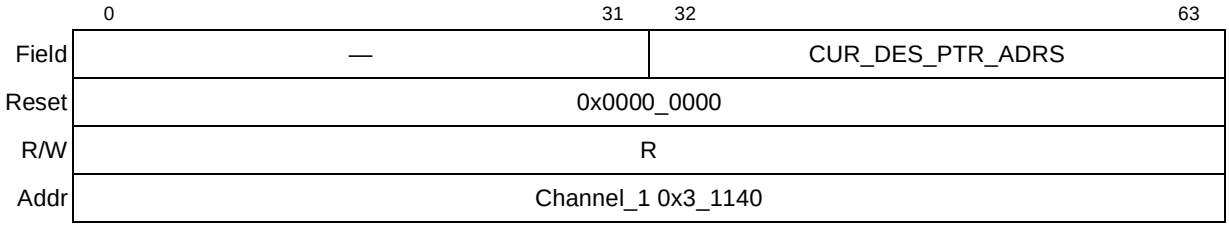


Figure 14-38. Crypto-Channel Current Descriptor Pointer Register (CDPR)

The bits in the CDPR perform the functions described in Table 14-37.

Table 14-37. CDPR Field Descriptions

Bits	Names	Description
0–31	—	Reserved, set to zero.
32–63	CUR_DES_PTR_ADRS	Current descriptor pointer address. Pointer to system memory location of the current descriptor. This field reflects the starting location in system memory of the descriptor currently loaded into the DB. This value is updated whenever the channel requests a fetch of a descriptor from the controller. The value from the fetch FIFO is transferred to the current descriptor pointer register immediately after the fetch is completed. This address will be used as the destination for writeback of the modified header dword, if header writeback notification is enabled.

14.5.1.4 Fetch FIFO (FF)

The channel contains a fetch FIFO to store a queue of pointers to descriptors that the channel will process.

The fetch FIFO, displayed in Figure 14-39, contains the addresses of the first byte of descriptors to be processed. In typical operation, the host CPU will create a descriptor in memory containing all relevant mode and location information for the SEC, then ‘launch’ the SEC by writing the address of the descriptor to the fetch FIFO.

The fetch FIFO can hold up to 24 descriptor pointers at a time. When the end of the current descriptor is reached, the descriptor pointed to by the next location in the fetch FIFO will be read to launch the next descriptor.

The Fetch Address is written into the FIFO only if the write includes the least significant byte (bits 56–63). If Extended Address mode is used, the Extended Fetch Address must be written before or concurrent with the Fetch Address.

Specifying a FETCH_ADRS of 0 causes the channel to generate an error and stop.

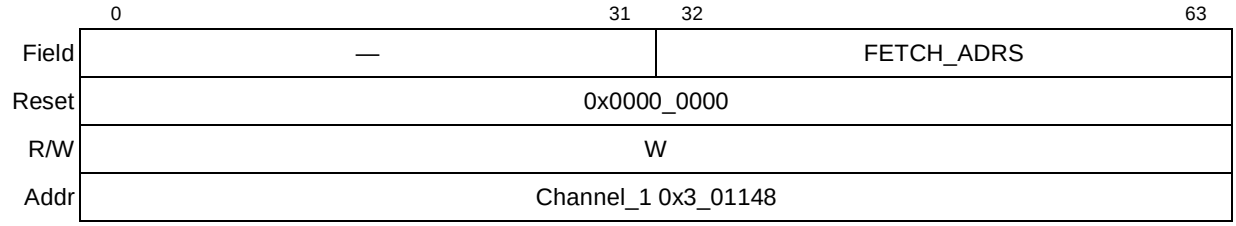


Figure 14-39. Fetch FIFO Register (FF)

Table 14-38 describes the fetch FIFO fields.

Table 14-38. Fetch FIFO Field Descriptions

Bits	Names	Description
0–31	—	Reserved, set to zero
32–63	FETCH ADRS	Fetch address. Pointer to system memory location of a descriptor the host wants the SEC to fetch.

14.5.1.5 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, because the descriptor is always fetched from system memory.

For more information about the fields in a descriptor, see Section 14.3.1, “Descriptor Structure.”

	0–15	16–17	18–23	24–27	28–31	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					

Figure 14-40. Descriptor Buffer (DB)

14.5.2 Channel Interrupts

The channel can assert both DONE and ERROR interrupts to the controller. When the interrupt generation conditions have been met, the channel will assert the appropriate interrupt. The status of the registered channel interrupts is available in the controller interrupt status register. The channel does not have an internal interrupt mask, but the controller can be programmed to block channel interrupts through its interrupt mask register (see Section 14.6.4.2, “Interrupt Mask Register (IMR)”).

14.5.2.1 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 14-36). Assuming the CDIE (Channel Done Interrupt Enable) is set, the channel will generate an interrupt event after every successfully completed descriptor (Notification Type set to Global), or after each successfully completed descriptor with the DN (Done Notification) bit set in the header word of the descriptor.

Even if multiple Channel Done interrupt events are generated by the channel before the first can be cleared by the host, the interrupt events are not lost. The controller queues channel done interrupts from the channel (see Section 14.6.3, “Controller Interrupts”).

14.5.2.2 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 14-35 for a complete listing of error types.

14.5.2.3 Channel Reset

Channel reset is asserted when the host sets the RESET bit in the channel configuration register (CCR). The effect of software reset on the channel varies according to what the channel is doing when the bit is set:

- If the RESET bit is set while the channel is requesting an EU assignment from the controller, the channel will cancel its request by asserting the release output signals. The channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.
- If the RESET bit is set after the channel has been dynamically assigned an 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 channel will then assert the appropriate release output signal to notify the controller that the channel has finished with the reserved EU(s). The channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.

14.6 Controller

The controller within the SEC is responsible for overseeing the operations of the execution units (EUs), the interface to the host processor, and the management of the 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:

- Provide arbitration for bus access and control bus accesses
- Control the internal bus accesses to the EUs
- Assign EUs to the channel
- Monitor interrupts from the channel and EUs and pass to host
- Realign read and write data to the proper byte alignment

14.6.1 Assignment of EUs to Channel

Assignment of a EU to the channel is done dynamically.[1] The channel requests an EU, the controller checks to see if the requested EU is available, and if it is, the controller grants the channel assignment of the EU.

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.

In some cases, the 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, the channel is then capable of requesting that the secondary EU snoop the bus. Snooping status is indicated in the MI and MO bits of Table 14-32.

In all cases, the controller assigns the primary EU to the requesting channel as the EUs become available. The controller does not wait until both EUs are available before issuing grants to the channel which is requesting two EUs.

14.6.2 Bus Interface

The controller in the SEC (refer to) has the ability to be a bus master or a slave. This means that the controller can issue read and write commands to the bus, and it can also be written to and read from by the host.

The controller is the sole bus master in the SEC. All other modules are slave-only devices. A channel may request access to system resources including the bus. In these cases, the channel must provide the starting address of the transfer for the bus(es) requested. All subsequent addresses are generated by the controller. All addresses will be sequential.

14.6.2.1 Arbitration for Use of the Controller and Buses

The controller attempts to maximize utilization of the system bus by grouping outstanding bus requests from the channel by request type (read or write). The controller will perform all write requests to the system bus, followed by all read requests, then repeat.

1. The security engine in the MPC8323E is a single-channel implementation, but other variants of the SEC have been implemented with 1, 2, and 4 channels. Although not necessary for a single-channel SEC, dynamic assignment of EUs to the channel is maintained to improve software compatibility with other SEC-enhanced processors.

The SEC does not dynamically adjust its own transaction priorities. System software, however, can adjust SEC transaction priority in realtime, with the change in priority taking effect immediately.

14.6.2.2 Master Read

Here is more detail on the sequence of events for an system bus read with the controller as master:

1. Channel asserts its bus read request to the controller.
2. Channel furnishes external read address, internal write address, and transfer length.
3. Controller sends request acknowledge to channel.
4. Controller asserts request to the system bus through the Magenta master interface.
5. Controller waits for system bus read to begin.
6. When bus read begins, controller receives data from the master interface and performs a write to the appropriate internal address supplied by the channel. Data may be realigned byte-wise by the controller if either:
   - The read 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.

When the SEC performs a transaction as master, it is possible for the intended slave to terminate the transfer due to an error. SEC transaction requests are posted to the MPC8323E target queue, after which the MPC8323E 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.

14.6.2.3 Master Write

Here is more detail on the sequence of events for an system bus write with the controller as master:

1. Channel asserts its bus write request to the controller.
2. Channel furnishes internal read address, external write address, and transfer length.
3. Controller sends request acknowledge to channel.
4. Controller performs a read from the appropriate internal address supplied by the channel, loads the write data into its FIFO, asserts a request to the system bus through the Magenta master interface, and waits for the system bus to become available.
5. When the system bus becomes available, controller writes data from its FIFO to the master interface.

14.6.3 Controller Interrupts

All interrupt outputs from other SEC blocks are fed to the controller as interrupt conditions. In addition, the controller itself detects some interrupt conditions. The controller maintains an interrupt status register

(ISR) with bits corresponding to all of these possible interrupt conditions. If an interrupt condition occurs and the corresponding bit of the interrupt mask register (IMR) is set, the associated ISR bit is set, indicating the presence of a pending interrupt. Whenever any bits are set in the interrupt status register, the controller asserts its interrupt output line to the host.

To handle an interrupt, the host must read the interrupt status register to determine the source. It may then need to do further reads of interrupt status registers of other blocks to get more detailed information about the cause. In some cases, the host may need to take action to clear the root cause of the interrupt. After that, the host can clear the desired bit of the interrupt status register by writing a 1 to the corresponding bit of the interrupt clear register (ICR). If the cause of the interrupt condition has not been cleared, or if there is some other interrupt condition from the same source, then the interrupt status register bit will clear for a cycle and go high again, and the interrupt output line to the host remains high. If the ISR bit is successfully cleared and no other interrupt conditions are present, the controller de-asserts its interrupt output. If any interrupts are still pending in the interrupt status register, the interrupt output remains asserted.

Note that EU interrupt conditions may be blocked at two different levels. There is an interrupt control register in each EU which can block particular interrupt conditions before they reach the EU's interrupt status register, and in addition, bits of the controller's interrupt mask register (IMR) must be set to allow interrupt conditions to reach the interrupt status register. Interrupt conditions from the channel and controller can only be blocked through the IMR.

For typical operation it is suggested that the IMR be programmed as follows: unmask channel interrupts while masking EU interrupts. Errors or Done signals coming from the EUs eventually cause the channel to signal an Error or Done interrupt.

The channel can generate frequent interrupts, especially if it is configured to interrupt at the completion of each descriptor. To make sure that the host receives the right number of interrupts, the channel Done interrupt has a special 'queueing' feature. If multiple Channel Done interrupts are generated before the first is cleared, then the additional interrupts are queued by the controller. When the host clears channel interrupt, if there are no other interrupts queued from that channel, then the channel Done interrupt is de-asserted. If other interrupts remain in the queue, the controller will de-assert the interrupt for one cycle and then re-assert it again.

14.6.4 Controller Registers

The controller registers are described in detail in the following sections.

14.6.4.1 EU Assignment Status Register (EUASR)

The EU assignment status register (EUASR), displayed in Figure 14-41, is used to check the assignment status of a EU to the channel.

A 1-bit field indicates to the channel whether or not the EU is assigned.

	0 3	4 7	8 11	12–14	15	16 19	20–22	23	24 27	28–30	31
Field	—	—	—	—	MDEU	—	—	AESU	—	—	DEU
Reset	0xF	0x0	0xF	0x0	0	0xF	0x0	0	0xF	0x0	0
R/W	R										
Addr	0x 3_1028										

	32 35	36 39	40 43	44 47	48 51	52 55	56 59	60 63
Field	—				—	—	—	—
Reset	0xFFF0				0xF	0x0	0xF	0x0
R/W	R							
Addr	0x3_ 102C							

Figure 14-41. EU Assignment Status Register (EUASR)

14.6.4.2 Interrupt Mask Register (IMR)

The SEC controller generates the single interrupt output from all possible interrupt sources. These sources can be individually enabled by the interrupt mask register (IMR). If unmasked, the interrupt source value, when active, is captured into the interrupt status register (ISR). Figure 14-42 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 disabled. The bit fields are described in Table 14-39.

For normal operation, the IMR should be programmed as follows: Unmask the channel interrupts while masking EU interrupts. The channels will generate the appropriate interrupts to the host.

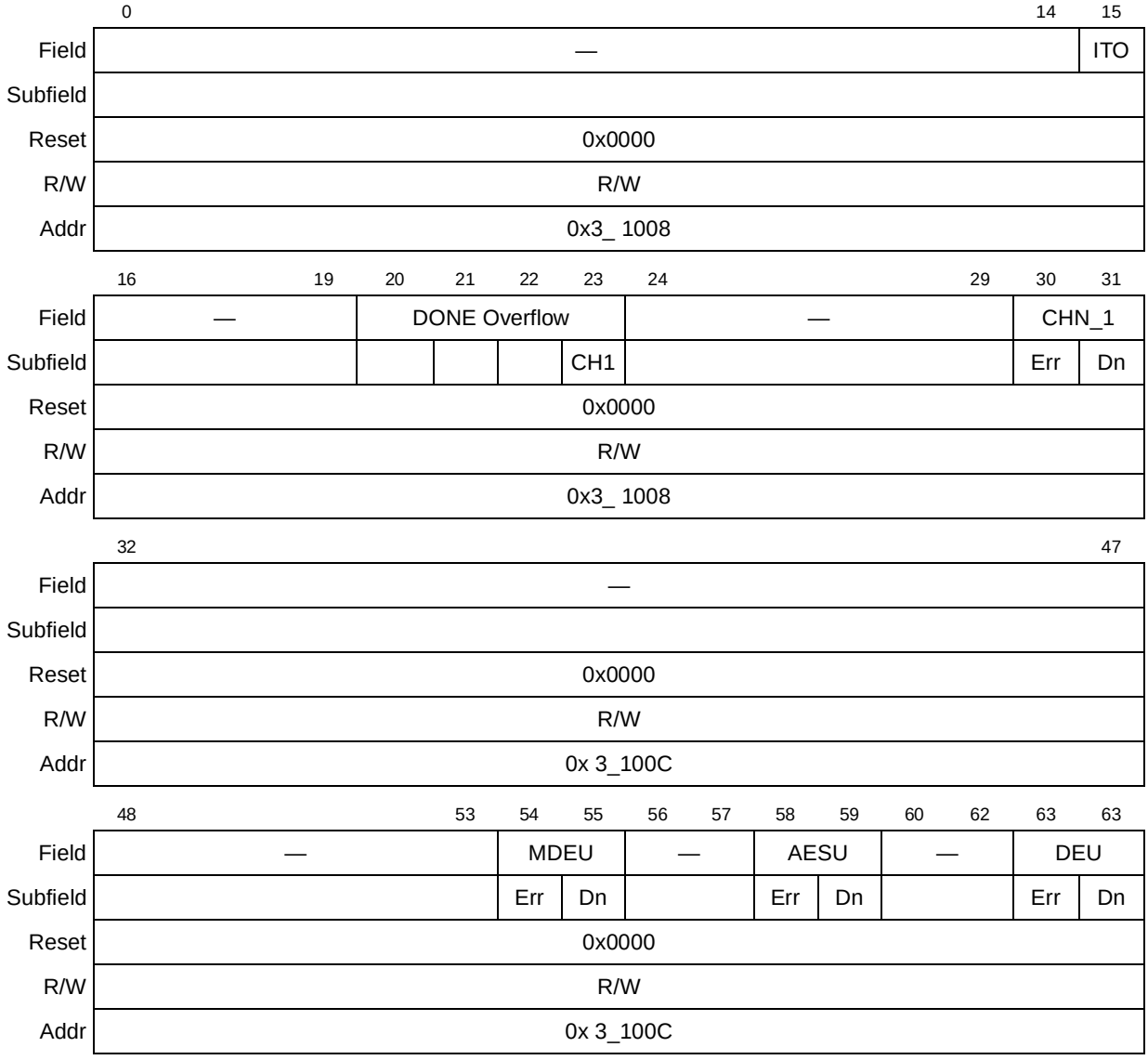


Figure 14-42. Interrupt Mask Register (IMR)

Table 14-39 describes the register field names in the interrupt mask register (IMR), interrupt status register (ISR), and interrupt clear register (ICR).

Table 14-39. Field Names in Interrupt Mask, Interrupt Status, and Interrupt Clear Registers

Bits	Names	Description
15	ITO	Internal time out 0 No internal time out 1 An internal time out was detected **Note:** Internal time out is an indication that the channel or EU has failed to respond to a slave read or write within 16 cycles, which would only occur in an impending hang condition. Assertion of this interrupt indicates the SEC controller has completed the transaction to avoid a hang, however the 'completed' transaction does not result in a successful read or write, and the interrupt advises the system that the slave transaction was unsuccessful.
20–23	Done Overflow	Done overflow (one bit for each channel) 0 No done overflow 1 Done overflow error. Indicates that more than 15 done interrupts were queued from the channel without an interrupt clear from the host.
30–31	Err and Dn bits for channel	Err 0 No error detected 1 Error detected. Indicates that channel status register must be read to determine exact cause of the error. Dn 0 Not DONE 1 DONE bit indicates that the interrupting channel has completed its operation.
Multiple	Err and Dn bits for execution units (AESU, and so on.)	Err 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. Dn 0 Not DONE. 1 DONE bit indicates that the interrupting EU has completed its operation.
0–14, 16–19, 24–29, 32–53, 56–57, 60–61	—	Reserved, set to zero.

14.6.4.3 Interrupt Status Register (ISR)

The interrupt status register (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 (ICR). Figure 14-43 shows the bit positions of each potential interrupt source. The bit fields are described in Table 14-39.

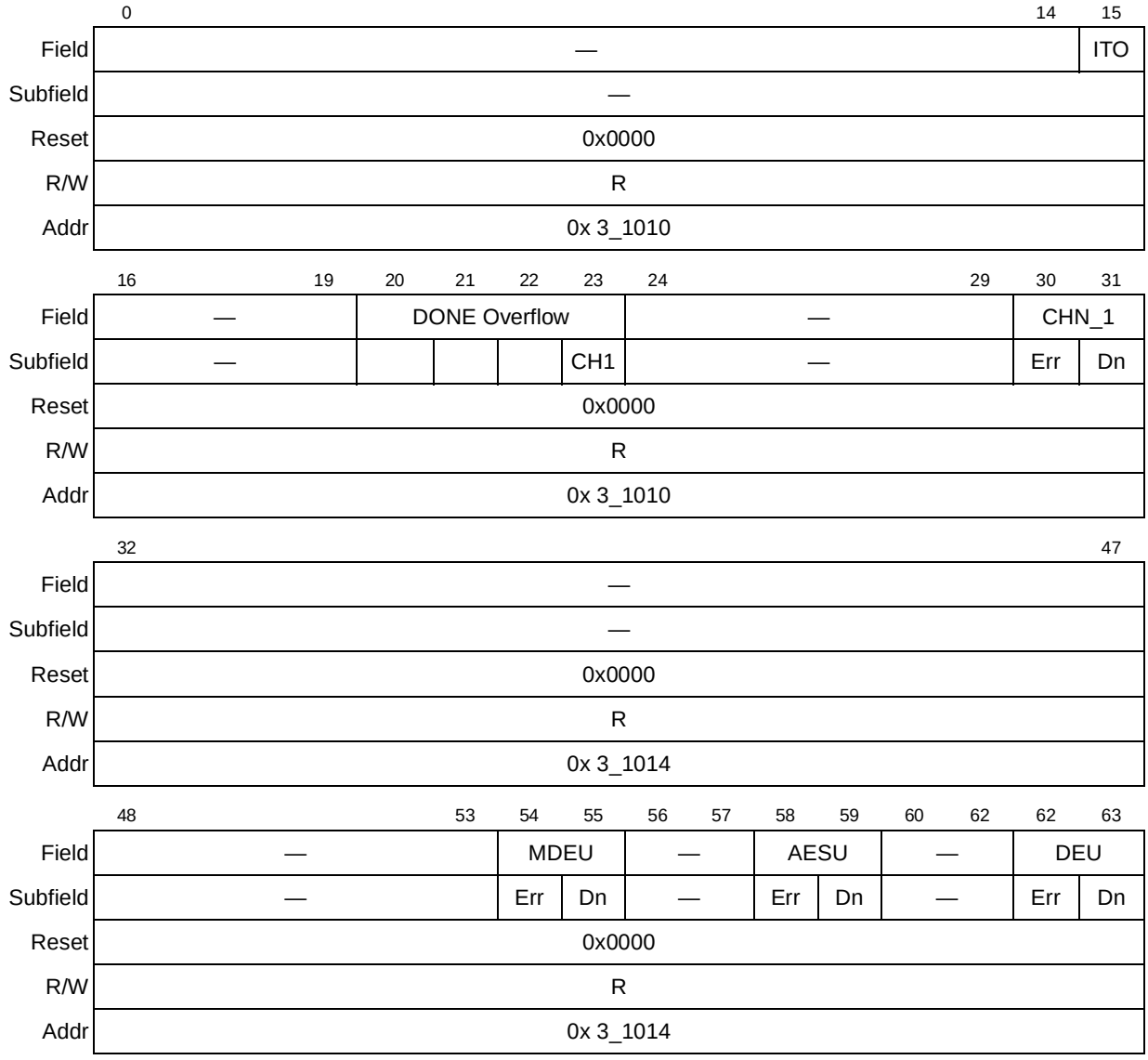


Figure 14-43. Interrupt Status Register (ISR)

14.6.4.4 Interrupt Clear Register (ICR)

The interrupt control register (ICR) provides a means of clearing the ISR. 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 14-44 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 14-44. The bit fields are described in Table 14-39.

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
Subfield			
Reset	0x0000		

Figure 14-44. Interrupt Clear Register (ICR)

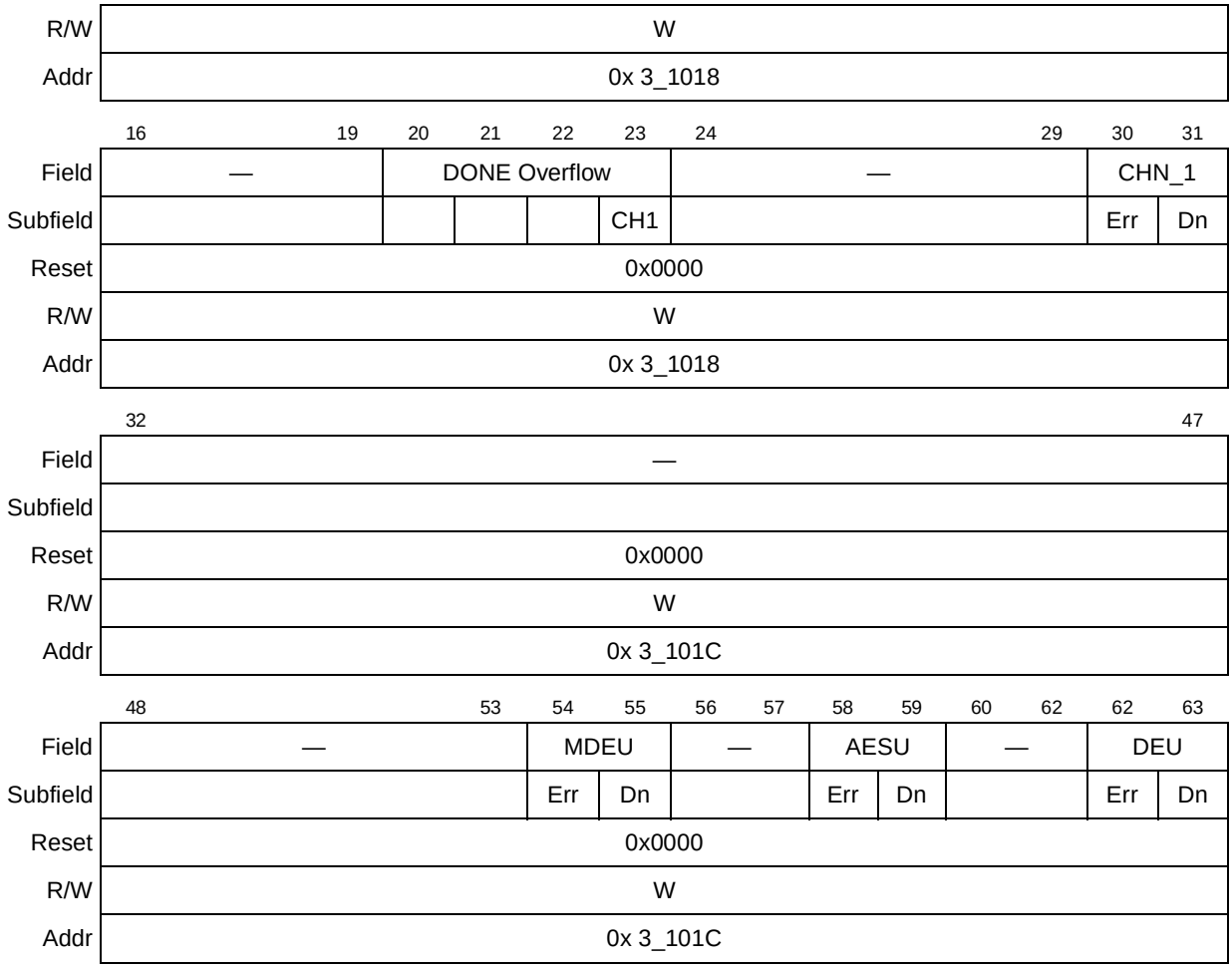


Figure 14-44. Interrupt Clear Register (ICR)

14.6.4.5 Identification Register (ID)

The read-only identification register (ID), displayed in Figure 14-45, contains a 64-bit value that uniquely identifies the version of SEC 2.2. The value of ID is always 0x0000_0000_0002_00A0, indicating that this is the first version of the SEC 2.2.

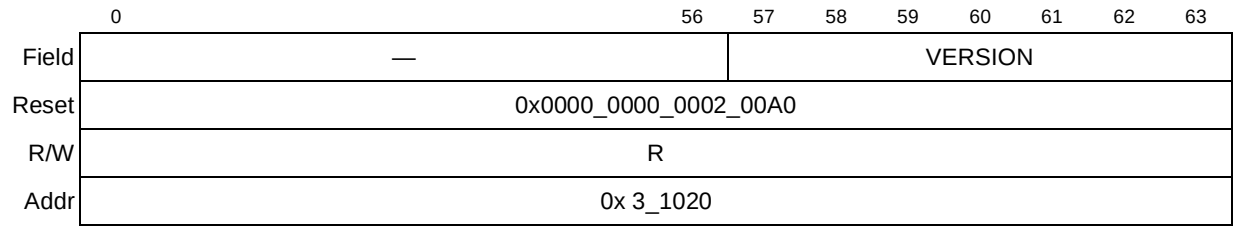


Figure 14-45. ID Register (ID)

14.6.4.6 IP Block Revision Register

The read-only IP block revision register, displayed in Figure 14-46, contains a 64-bit value that uniquely identifies the version of SEC 2.2. The value of this register is always 0x0000_0000_0002_00A0, indicating that this is the first version of SEC2.2.

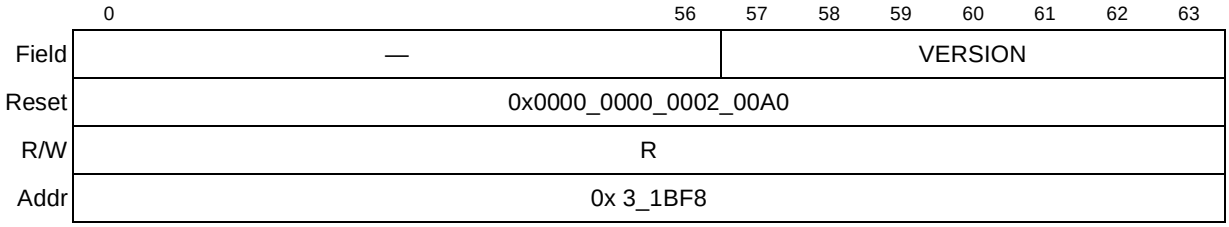


Figure 14-46. IP Block Revision Register

14.6.4.7 Master Control Register (MCR)

The master control register (MCR), shown in Figure 14-47, controls certain functions in the controller and provides a means for software to reset the SEC.

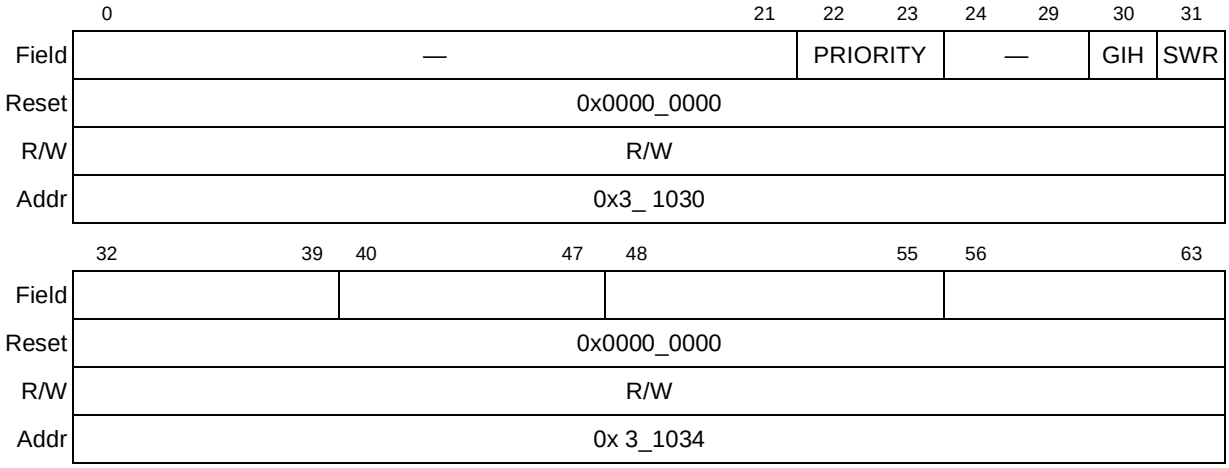


Figure 14-47. Master Control Register (MCR)

Table 14-40 describes the MCR fields.

Table 14-40. MCR Field Descriptions

Bits	Names	Description
0–21	—	Reserved
22–23	Priority	Priority on master bus. The setting of these bits determines the transaction priority level the SEC asserts to the device internal arbiter. The SEC does not dynamically alter its priority level based on system congestion or SEC utilization, however software may change the SEC priority level in realtime. 00 Lowest priority (default) 01 Next lowest priority 10 Next highest priority 11 Highest priority
24–29	—	Reserved

Table 14-40. MCR Field Descriptions (continued)

Bits	Names	Description
30	GIH	Global inhibit. Writing 1 to this bit prevents output IPM_SNOOP from being asserted. 0 Permit assertion of IPM_SNOOP, allowing the system cache to snoop bus transactions initiated by the SEC. 1 Prevent assertion of IPM_SNOOP, preventing the system cache from snooping
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 Do not reset 1 Global reset

14.6.5 Snooping by Caches

SEC transactions can be snooped by the MPC8323E cache if defined as global. This definition is programmed in the master control register MCR[GI]. See 14.6.4.7, “Master Control Register (MCR),” for more details. Note that SEC transactions are defined as global by default.

14.6.6 Interrupts

The SEC generates a single interrupt to the MPC8323E programmable interrupt controller. The user allows interrupts from the SEC to be reported to the CPU by setting the mask bit in SIMSR_H[SEC].

The user can control which events cause an interrupt by configuring the SEC interrupt mask register (IMR). 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 write 1 to the bits in the SEC interrupt clear register (ICR) corresponding to the pending ISR bits.

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.

14.7 Power Saving Mode

The SEC can be disabled by clearing SCCR[ENCCM]. See Section 4.5.2.3, “System Clock Control Register (SCCR),” for more information.

Chapter 15
I²C Interface

This chapter describes the inter-IC (IIC or I²C) bus interface implemented on this device.

15.1 Introduction

The inter-IC (IIC or I²C) bus is a two-wire—serial data (SDA) and serial clock (SCL)—bidirectional serial bus that provides a simple, efficient method of data exchange between this device and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. Figure 15-1 shows a block diagram of the I²C interface.

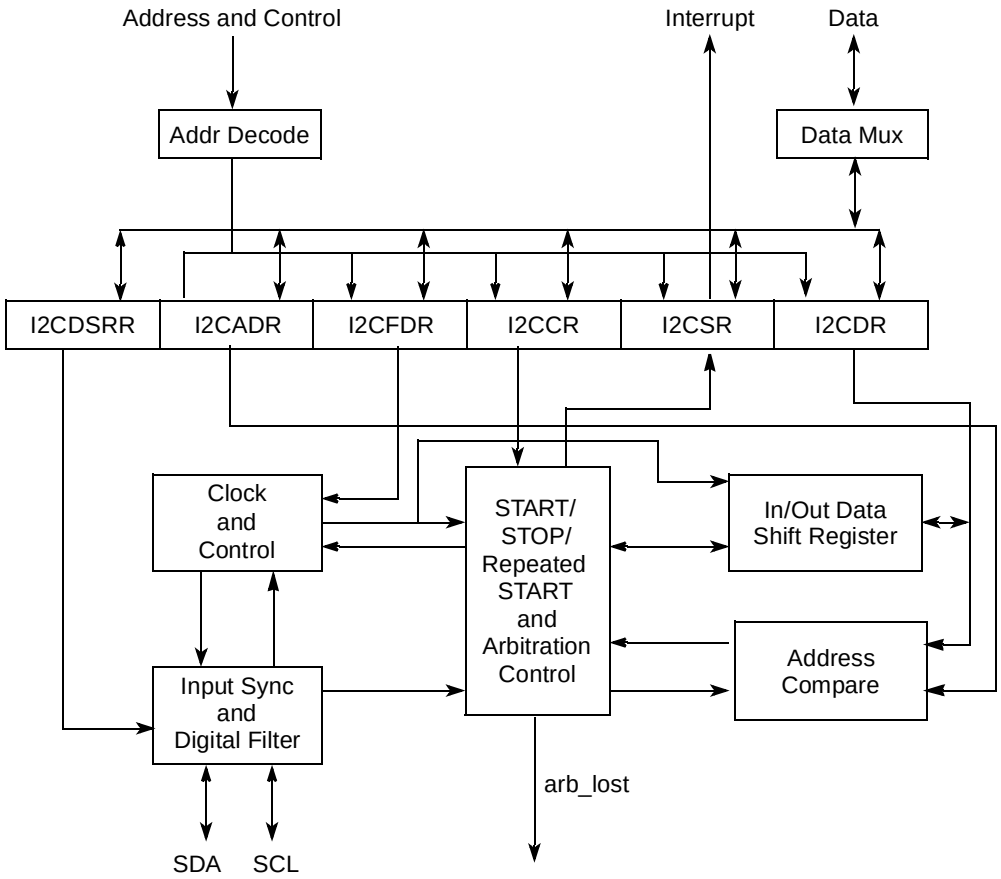


Figure 15-1. I²C Block Diagram

The two-wire I²C bus minimizes interconnections between devices. The synchronous, multiple-master I²C bus allows the connection of additional devices to the bus for expansion and system development. The bus includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

15.1.1 Features

The I²C interface includes the following features:

- Two-wire interface
- Multiple-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

15.1.2 Modes of Operation

The I²C unit on this device can operate in one of the following modes:

- Master mode. The I²C initiates a transfer, generates clock signals, and terminates a transfer. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode. The I²C is addressed by an I²C master. The module must be enabled before a START condition from an I²C master is detected.
- Interrupt-driven byte-to-byte data transfer. When successful slave addressing is achieved (and SCL 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. I²C controller supports boot sequencer mode. This mode can be used to initialize the configuration registers in the device after the I²C module is initialized. Boot sequencer mode is selected using the BOOTSEQ field in the reset configuration word high. Note that the hard-coded reset configuration word high value is boot sequencer mode disabled.
- Reset configuration load. In this mode, the I²C interface loads the reset configuration words from an EEPROM at a specific calling address while the rest of the device is in the reset state ($\overline{\text{HRESET}}$ asserted). Once the reset configuration words are latched inside the device, I²C is reset until $\overline{\text{HRESET}}$ is negated. After $\overline{\text{HRESET}}$ is negated, the device may be initialized using boot sequence mode according to the BOOTSEQ field in the reset configuration word. See Section 15.4.5, “Boot Sequencer Mode.”

Additionally, the following three I²C–specific states are defined for the I²C interface:

- START condition. This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves.

- Repeated START condition. A START condition that is generated without a STOP condition to terminate the previous transfer.
- STOP condition. The master can terminate the transfer by generating a STOP condition to free the bus.

15.2 External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

15.2.1 Signal Overview

The I²C interface uses the SDA and SCL signals, described in Table 15-1, 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 15-1. I²C Interface Signal Descriptions

Signal Name	Idle State	I/O	State Meaning
Serial Clock (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 negates for data pacing.
Serial Data (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.

15.2.2 Detailed Signal Descriptions

SDA and SCL, described in Table 15-2, serve as a communication interconnect with other devices. All devices connected to these signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the hardware specifications for electrical characteristics.

Table 15-2. I²C Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
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*n* as the clock for the data.
	I	As inputs for the bi-directional 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.

Table 15-2. I²C Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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 bi-directional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated—Data is driven.
	I	As inputs for the bi-directional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated—Used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

15.3 Memory Map/Register Definition

Table 15-3 lists the I²C–specific registers and their addresses.

Table 15-3. I²C Memory Map

Address	I²C Register	Access	Reset	Section/Page
0x0_3000	I2CADR—I²C address register	R/W	0x00	15.3.1.1/15-5
0x0_3004	I2CFDR—I²C frequency divider register	R/W	0x00	15.3.1.2/15-5
0x0_3008	I2CCR—I²C control register	R/W	0x00	15.3.1.3/15-6
0x0_300C	I2CSR—I²C status register	R/W	0x81	15.3.1.4/15-7
0x0_3010	I2CDR—I²C data register	R/W	0x00	15.3.1.5/15-9
0x0_3014	I2CDFSRR—I²C digital filter sampling rate register	R/W	0x10	15.3.1.6/15-9
0x0_301C–0x0_31FF	Reserved, should be cleared	—	—	—

15.3.1 Register Descriptions

This section describes the I²C registers in detail. Note that reserved bits should always be written with the value they return when read. That is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. The return value of the reserved fields should not be assumed, even though the reserved fields return zero. This does not apply to the I²C data register (I2CDR).

15.3.1.1 I²C Address Register (I2CADR)

Figure 15-2 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.

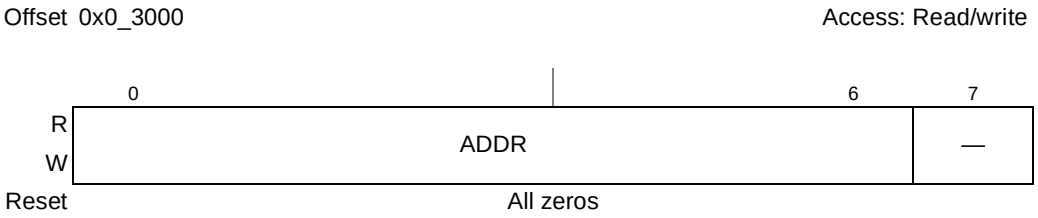


Figure 15-2. I²C Address Register (I2CADR)

Table 15-4 describes the bit settings of I2CADR.

Table 15-4. 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, should be cleared

15.3.1.2 I²C Frequency Divider Register (I2CFDR)

Figure 15-3 shows the bits of the I²C frequency divider register.

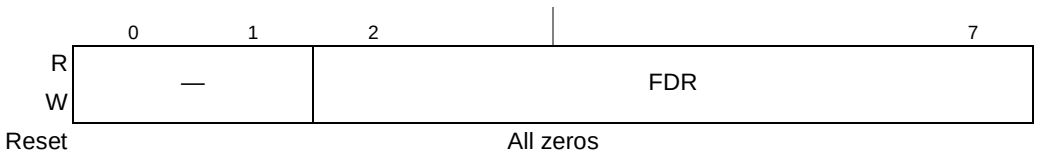


Figure 15-3. I²C Frequency Divider Register (I2CFDR)

Table 15-5 describes the bit settings of I2CFDR. It also maps I2CFDR[FDR] to the clock divider values. Although it describes the ratio between the I²C controller internal clock and SCL, the default ratio of I²C controller clock and CSB is 1:1. This ratio is set in SCCR[ENCCM]. Clock ratios of I²C1 are controllable but clock ratio for I²C2 is not and it is always 1:1 with CSB. Consider this factor when selecting an FDR value.

Table 15-5. I2C FDR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared
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 I²C controller clock divided by the divider. 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 **Note:** The value's shown in the table are applicable only for the default value of DFSRR. Refer to AN2919. **Note:** I2C controller clock of I2C1 is derived from csb_clk / SCCR[SDHCCM].

15.3.1.3 I²C Control Register (I2CCR)

Figure 15-4 shows the I²C control register.

Offset 0x0_3008 Access: Mixed

	0	1	2	3	4	5	6	7
R	MEN	MIEN	MSTA	MTX	TXAK		—	BCST
W						RSTA		
Reset	All zeros							

Figure 15-4. I²C Control Register (I2CCR)

Table 15-6 describes the I2CCR bit settings.

Table 15-6. I2CCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. Controls the software reset of the I²C module. 0 The module is reset and disabled. The interface is held in reset, but the registers can still be accessed. 1 The I²C module is enabled. MEN 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 On a transition to zero, a STOP condition is generated and the mode changes from master to slave. Cleared without generating a STOP condition when the master loses arbitration. 1 When MSTA changes from zero to one, a START condition is generated on the bus and master mode is selected.
3	MTX	Transmit/receive mode select. Selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to 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 high. MTX is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. Specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit applies only 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 bit after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
5	RSTA	Repeated START. Note that this bit is not readable, which means if a read is performed to RSTA, a zero value is returned. 0 No START condition is generated 1 Setting this bit always generates a repeated START condition on the bus, provides the device with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration.
6	—	Reserved, should be cleared
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I²C to accept broadcast messages at address zero

15.3.1.4 I²C Status Register (I2CSR)

I2CSR is shown in Figure 15-5.

Offset 0x0_300C 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 15-5. I²C Status Register (I2CSR)

Table 15-7 describes the bit settings of the I2CSR.

Table 15-7. 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 I2CDR is read in receive mode or when I2CDR is written in transmit mode. • After a start sequence is recognized by the I²C controller in slave mode. 1 Byte transfer is completed
1	MAAS	Addressed as a slave. When the value in I2CADR matches the calling address or when the calling address is the broadcast address and broadcast mode is enabled (I2CCR[BCST] is set), 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. Writing to the I2CCR automatically clears this bit. 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address and broadcast mode is enabled. This is also set if this I²C drives an address of all 0s.
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 SRW, the processor can select slave transmit/receive mode according to the command of the master.
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided 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*n* during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

15.3.1.5 I²C Data Register (I2CDR)

The I2C data register is shown in Figure 15-6.

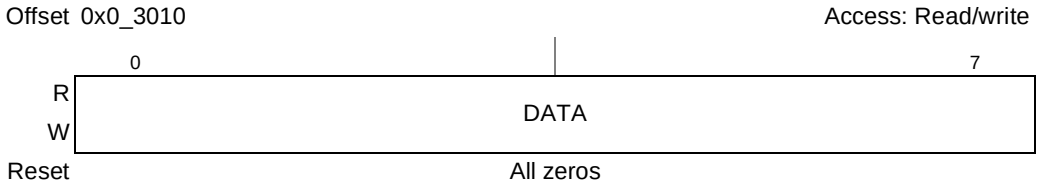


Figure 15-6. I²C Data Register (I2CDR)

Table 15-8 shows the bit descriptions for I2CDR.

Table 15-8. 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 in both master receive and slave receive modes, the very first read is always a dummy read.

15.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

I2CDFSRR is shown in Figure 15-7.

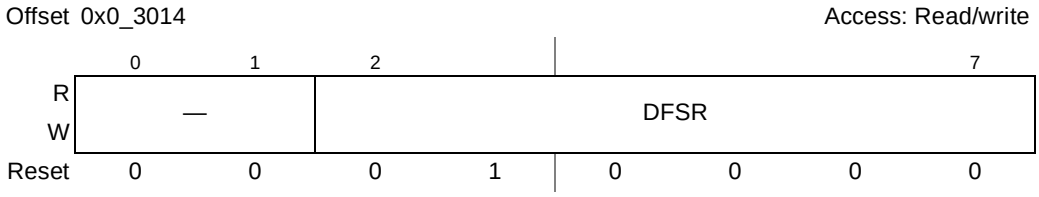


Figure 15-7. I²C Digital Filter Sampling Rate Register (I2CDFSRR)

Table 15-9 shows the I2CDFSRR field descriptions.

Table 15-9. I2CDFSRR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared
2–7	DFSR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. DFSR 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 frequency by the non-zero value of DFSR. If I2CDFSRR is cleared, the I²C bus sample points default to the reset divisor 0x10.

15.4 Functional Description

The I²C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. If boot sequencer mode is selected, the I²C interface performs as a slave receiver after the boot sequence has completed.

15.4.1 Transaction Protocol

A standard I²C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 15-8 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I²C protocol. The details of the protocol are described in the following subsections.

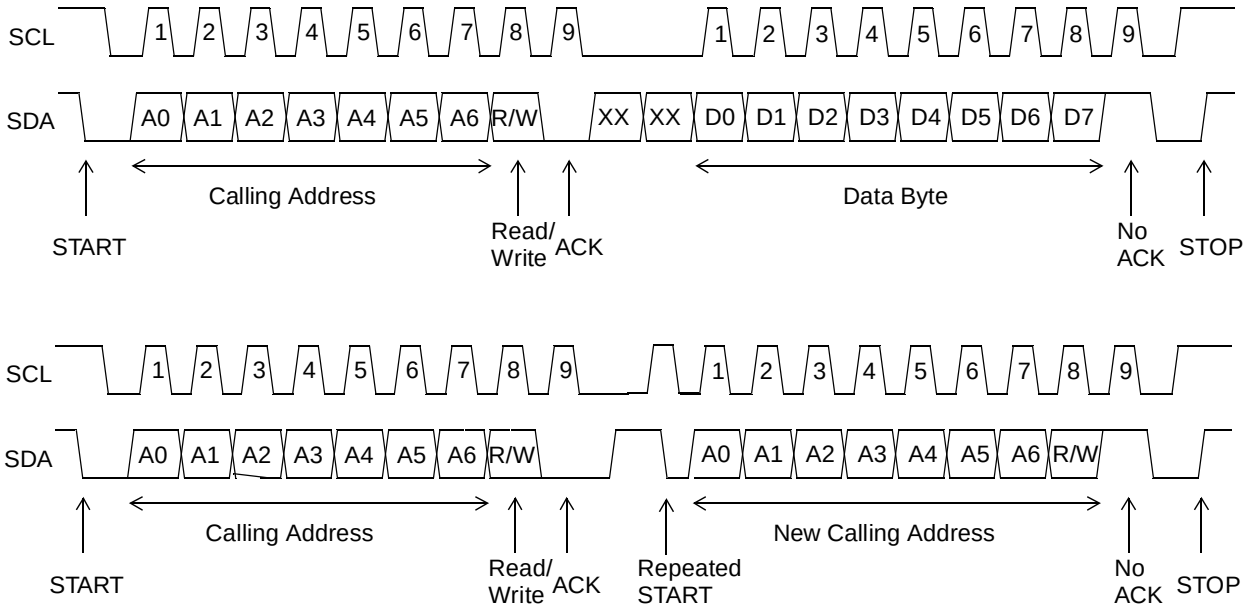


Figure 15-8. I²C Interface Transaction Protocol

15.4.1.1 START Condition

When the I²C bus is not engaged (both SDA and SCL lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 15-8, a START condition is defined as a high-to-low transition of SDA 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].

15.4.1.2 Slave Address Transmission

The first byte of data transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/$\overline{W}$ bit, which indicates the direction of the data transferred to the slave. Each slave in the system has a unique address. When the I²C module is operating as a master, it must not transmit an address that is the same as its slave address. An I²C device cannot be master and slave at the same time.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (negating the SDA signal at the 9th clock) as shown in Figure 15-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL 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 does not check the R/W bit. The second byte of the broadcast message is the master address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to 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 15-8. 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 SDA 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.

15.4.1.3 Repeated START Condition

Figure 15-8 shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

15.4.1.4 STOP Condition

The master can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the SDA signal while SCL is high. For more information, see Figure 15-8. Note that a master can generate a STOP even if the slave has transmitted an acknowledge bit, at which point the slave must release the bus. The STOP condition is initiated by a software write that clears I2CCR[MSTA].

As described in Section 15.4.1.3, “Repeated START Condition,” the master can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

15.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in the I²C module.

15.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I²C data transfers are monitored as follows (see Figure 15-8):

- START conditions are detected when an SDA 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.

15.4.1.5.2 Control Transfer—Implementation Details

The I²C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I²C. The SCL output is pulled low as determined by the internal clock generated in the clock module. The SDA output can change only at the midpoint of a low cycle of the SCL, unless it is performing a START, STOP, or repeated START condition. Otherwise, the SDA output is held constant.

SDA is negated when one or more of the following conditions are true:

- Master mode
  - Data bit (transmit)
  - ACK bit (receive)
  - START condition
  - STOP condition
  - Repeated START condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - ACK bit (receive)

The SCL 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
  - Repeated START condition begin
  - Repeated START condition end

- Slave mode
  - Address cycle
  - Transmit cycle
  - ACK cycle

15.4.1.6 Address Compare—Implementation Details

The address compare block determines whether a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The following address comparisons are performed:

- Whether a broadcast message has been received, to update I2CSR
- Whether the module has been addressed as a slave, to update I2CSR and to generate an interrupt
- Whether the address transmitted by the current master matches the general broadcast address

15.4.2 Arbitration Procedure

The I²C interface is a true multiple-master bus. If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I²C module) determines the bus clock—the low period is equal to the longest clock-low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA 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 I2CSR[MAL] to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

If the I²C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—the I²C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—the I²C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately causes in the current bus master to lose arbitration, after which bus operations return to normal.

15.4.2.1 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA line while attempting to drive a 1, tries to generate a START or repeated START at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and 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 repeated START condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

Note that the I²C module does not automatically retry a failed transfer attempt.

15.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL 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.

15.4.4 Clock Control

The clock control block handles requests from the clock signal for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
  - Transmit slave address after START condition
  - Transmit slave address after repeated START condition
  - Transmit data
  - Receive data
- Slave mode
  - Transmit data
  - Receive data
  - Receive slave address after START or repeated START condition

15.4.4.1 Clock Synchronization

Due to the wire AND logic on the SCL 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 negates the SCL line. After a device has negated SCL, 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 SCL if another device is still within its low period. Therefore, 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 periods, SCL is released and asserted. Then there is no difference between the devices' clocks and the state of SCL, and all the devices begin counting their high periods. The first device to complete its high period negates SCL again.

15.4.4.2 Input Synchronization and Digital Filter

The following sections describes synchronization of the input signals and the filtering of SCL and SDA in detail.

15.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

15.4.4.2.2 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 I2CDFSRR to control the filtered sampling rate.

15.4.4.3 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL 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, the resulting SCL low period is extended.

15.4.5 Boot Sequencer Mode

Boot sequencer mode is selected at power-on reset by the BOOTSEQ field of the high-order reset configuration word. If boot sequencer mode is selected, the I²C module communicates with one or more EEPROMs through the I²C interface. EEPROMs can be programmed to initialize one or more configuration registers. Note that as described in Section 4.3.2.2.3, “Boot Sequencer Configuration,” the default value for BOOTSEQ is 0b00, which corresponds to the I²C boot sequencer being disabled at power-up.

Boot sequencer mode also supports an extension of the standard I²C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes. This extended addressing mode is selectable using a different encoding in the BOOTSEQ field of the high-order reset configuration word (see Section 4.3.2.2.3, “Boot Sequencer Configuration.”) In this mode, only one EEPROM device can be used and the maximum number of registers is limited by the size of the EEPROM.

If the standard I²C interface is used, the I²C module addresses the first EEPROM, and reads 256 bytes. Then it issues a repeated start and addresses the next EEPROM address. This sequence continues until the CONT bit is cleared. If the last register is not detected before wrapping back to the first address, an error condition is detected. In other words, if the CONT bit for not cleared on the final 7 bytes, an error condition is detected, causing the I²C controller to hang. The I²C module continues to read from the EEPROM as long as the continue (CONT) bit is set in the EEPROM. The CONT bit resides in the address/attributes field that is transferred from the EEPROM, as described in Section 15.4.5.2, “EEPROM Calling Address.” There should be no other I²C traffic when the boot sequencer is active.

15.4.5.1 Using the Boot Sequencer for Reset Configuration

The reset configuration word can be loaded by using the I²C boot sequencer. See Section 4.3.2.2.3, "Boot Sequencer Configuration."

Note that this usage does not prevent using the I²C boot sequencer to initiate the device in the normal functional mode, after reset state has completed. However, an I²C serial EEPROM of extended addressing type must be used and the first two EEPROM data structures must contain dedicated reset information.

15.4.5.2 EEPROM Calling Address

The EEPROM calling address is 0b101_0000. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. Any additional EEPROMs are addressed in sequential order.

15.4.5.3 EEPROM Data Format

The I²C module expects a particular format for data to be programmed in the EEPROM. Figure 15-9 shows an example of the EEPROM contents, including the preamble, data format, and CRC.

0	1	2	3	4	5	6	7	
1	0	1	0	1	0	1	0	Preamble
0	1	0	1	0	1	0	1	
1	0	1	0	1	0	1	0	
ACS	BYTE_EN				1	ADDR[12:13]		First Configuration Preload Command
ADDR[14:21]								
ADDR[22:29]								
DATA[0:7]								
DATA[8:15]								
DATA[16:23]								
DATA[24:31]								
ACS	BYTE_EN				1	ADDR[12:13]		Second Configuration Preload Command
ADDR[14:21]								
ADDR[22:29]								
DATA[0:7]								
DATA[8:15]								
DATA[16:23]								
DATA[24:31]								
.....								
ACS	BYTE_EN				1	ADDR[12:13]		Last Configuration Preload Command
ADDR[14:21]								
ADDR[22:29]								
DATA[0:7]								
DATA[8:15]								
DATA[16:23]								
DATA[24:31]								
0	0	0	0	0	0	0	0	End Command
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
CRC[0:7]								
CRC[8:15]								
CRC[16:23]								
CRC[24:31]								

Figure 15-9. EEPROM Contents

- A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I²C checks to ensure that this preamble is correctly detected before proceeding.
- Following the preamble, there should be a series of configuration registers (known as register preloads). Each configuration register should be programmed according to a particular format, as shown in Figure 15-10.

0	1	2	3	4	5	6	7
ACS	BYTE_EN				CONT	ADDR[12:13]	
ADDR[14:21]							
ADDR[12:29]							
DATA[0:7]							
DATA[8:15]							
DATA[16:23]							
DATA[24:31]							

Figure 15-10. EEPROM Data Format for One Register Preload Command

  - The first byte holds alternate configuration space (ACS), byte enables, and continue (CONT) attributes.
  - The 2 least-significant bits of the address are derived from the byte enables. address offset. Therefore, the address offset programmed into the EEPROM preload should be a word offset.
  - The most significant 16 bits (assuming 36-bit addressing) of the address are prepended from either IMMRRBAR or alternate configuration space.
  - After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of transaction.

Byte enables should be asserted for any byte that will be written, and they should be asserted contiguously, creating a 1, 2, or 4 byte write to a register. The boot sequencer assumes that a big-endian address is stored in the EEPROM. In addition, byte enable bit 0 (bit 1 of the byte) corresponds to the most-significant byte of data (data[0:7]), and byte enable bit 3 (bit 4 of the byte) corresponds to the least-significant byte of data (data[24:31]).

By asserting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer according to the value in the ALTCBAR register. This will allow for external memories to be configured. Otherwise, IMMRBAR is prepended to the EEPROM address.

If the CONT bit is cleared, the first 3 bytes, including ACS, the byte enables, and the address, should be cleared 0. Also, the data contains the final CRC. A CRC-32 algorithm is used to check the integrity of the data. The following polynomial is used:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

The CRC should cover all bytes stored in the EEPROM before the CRC. This includes the preamble, all register preloads, and the first 3 bytes of the last 7-byte preload (which should be all zeros).

15.4.5.4 Boot Sequencer Done Indication

Dedicated hardware is not provided to indicate whether the boot sequencer operation completed successfully. It is recommended to use one of the GPIO signals for that purpose. To do this, the last register preload programmed into the EEPROM should contain the address of the appropriate GPIO register and data that causes the setting of the required GPIO signal. The GPIO signal may be used for an external device or for debug purposes.

15.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. Figure 15-11 is a recommended flowchart for I²C interrupt service routines.

A **sync** assembly instruction must be executed after each I²C register read/write access to guarantee that register accesses occur in order.

The I²C controller does not guarantee its recovery from all illegal I²C bus activity. In addition, a malfunctioning device may hold the bus captive. A good programming practice is for software to rely on a watchdog timer to help recover from I²C bus hangs. The recovery routine should also handle the case when the illegal I²C bus behavior causes the status bits returned after an interrupt to be inconsistent with what was expected.

15.5.1 Interrupt Service Routine Flowchart

Figure 15-11 shows an example algorithm for an I2C interrupt service routine. Deviation from the flowchart may result in unpredictable I2C bus behavior. However, in the slave receive mode (not shown), the interrupt service routine may need to set I2CCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that a **sync** instruction follow each I2C register read or write to guarantee that register accesses occur in order.

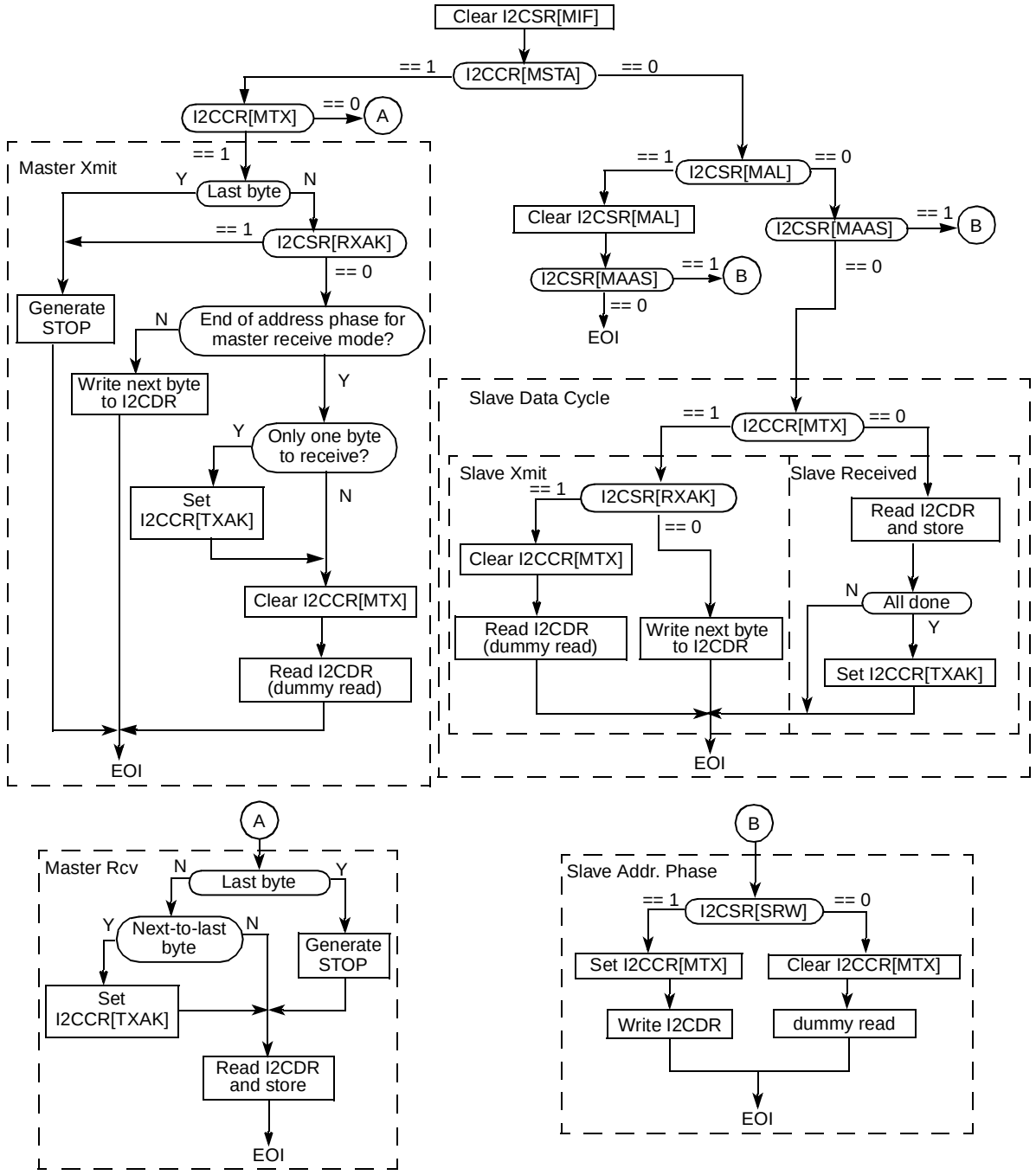


Figure 15-11. Example I2C Interrupt Service Routine Flowchart

15.5.2 Initialization Sequence

A hard reset initializes all of the I²C registers to their default states. The following initialization sequence initializes the I²C unit:

1. All I²C registers must be located in a cache-inhibited page.
2. Update I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the CSB (platform) clock.
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.

15.5.3 Generation of START

After initialization, the following sequence can be used to generate START:

1. If the device is connected to a multimaster I²C system, check whether the serial bus is free (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).

15.5.4 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 I2CDR in receive mode or write to I2CDR in transmit mode. Note that this causes I2CSR[MCF] to be cleared, as shown in Figure 15-11.
3. When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CCR[MTX] must be toggled at this stage (see Figure 15-11).

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 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 Figure 15-11).

15.5.5 Generation of STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can generate a STOP condition after all the data has been transmitted.

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data (by setting the transmit acknowledge bit (I2CCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has been transferred on the I²C interface, so the last byte does not receive the data acknowledge (because I2CCR[TXAK] is set). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

15.5.6 Generation of Repeated START

At the end of a data transfer, if the master still wants to communicate on the bus, it can generate another START condition followed by another slave address without first generating a STOP condition. This is accomplished by setting I2CCR[RSTA].

15.5.7 Generation of SCL When SDA is Negated

It is sometimes necessary to force the I²C module to become the I²C bus master out of reset and drive SCL*n* (even though SDA 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 negated low by another I²C device while this I²C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force this I²C module to generate SCL 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 I2CDR.
4. Return the I²C module to slave mode by setting I2CCR to 0x80.

15.5.8 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the module addressed as a slave should be tested to check if a calling of its own address has been received. If 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 negates SCL between byte transfers. SCL is released when I2CDR is accessed in the required mode.

15.5.8.1 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 Figure 15-11.

15.5.8.2 Loss of Arbitration and Forcing of Slave Mode

When a master loses arbitration the following conditions all occur:

- 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 15.4.2.1, “Arbitration Control.”

Chapter 16
DUART

This chapter describes the two (dual) universal asynchronous receiver/transmitters (UARTs) of the device. It describes the functional operation, the DUART initialization sequence, and the programming details for the DUART registers and features.

16.1 Overview

The DUART consists of two (dual) universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the system clock. The DUART programming model is compatible with the PC16552D.

The UART interface is point-to-point, meaning that only two UART devices are attached to the connecting signals. As shown in Figure 16-1, each UART module consists of the following:

- Receive and transmit buffers
- Clear to send ($\overline{\text{CTS}}$) input port and request to send ($\overline{\text{RTS}}$) output port for data-flow control.
- 16-bit counter for baud rate generation
- Interrupt control logic

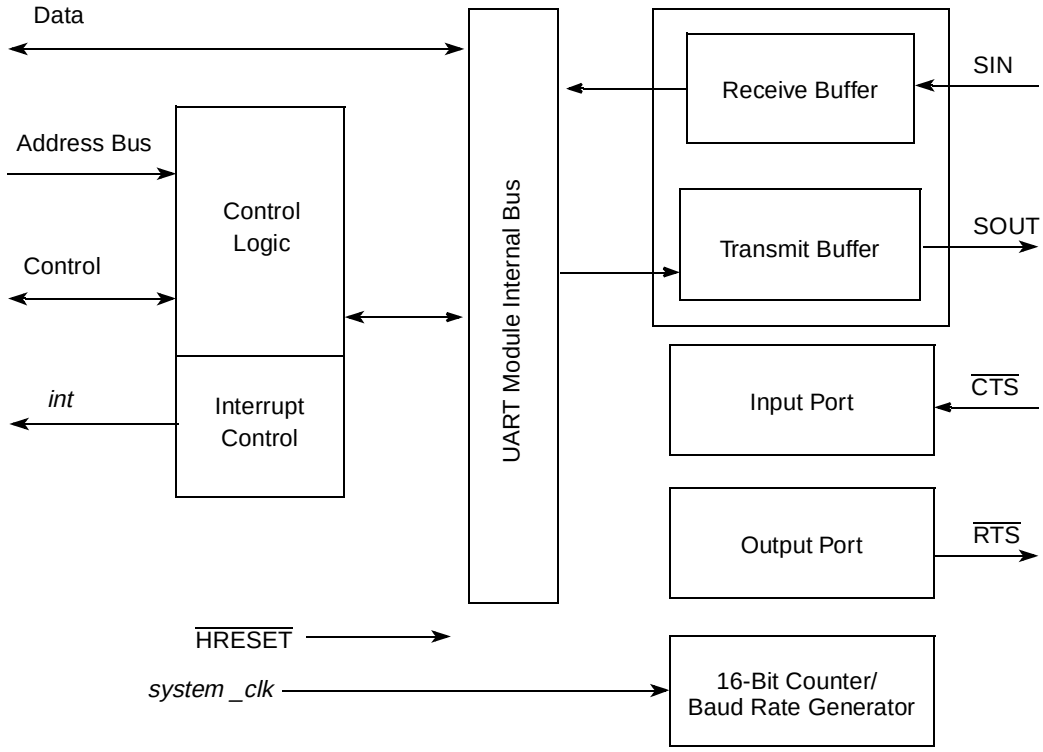


Figure 16-1. UART Block Diagram

16.1.1 Features

The DUART includes these features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and MODEM status interrupts
- Software-programmable baud generators that divide the system clock by 1 to (2^{16}–1) and generate a 16x clock for the transmitter and receiver engines
- Clear-to-send ($\overline{\text{CTS}}$) 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

16.1.2 Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the system clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream, inserting the appropriate START, STOP, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a START bit, parity (if any), STOP bits, and transfers the assembled character (with START, STOP, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

16.2 External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

16.2.1 Signal Overview

Table 16-1 summarizes the DUART signals. Note that although the actual device signal names are prepended with the 'UART_' prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

Table 16-1. DUART Signal Overview

Signal Name	I/O	Pins	Reset Value	State Meaning
UART_SIN[1:2]	I	2	1	Serial in data UART1 and UART2
UART_SOUT[1:2]	O	2	1	Serial out data UART1 and UART2
$\overline{\text{UART\_CTS}}$[1:2]	I	2	1	Clear to send UART1 and UART2
$\overline{\text{UART\_RTS}}$[1:2]	O	2	1	Request to send UART1 and UART2

16.2.2 Detailed Signal Descriptions

The DUART signals are described in detail in Table 16-2.

Table 16-2. DUART Signals—Detailed Signal Descriptions

Signal	I/O	Description	
UART_SIN[1:2]/DSP_UART_SIN	I	Serial data in. Data is received on the receivers of UART1, UART2, or DSP_UART through its respective serial data input signal, with the least significant bit received first.	
		State Meaning	Asserted/Negated—Represents the data being received on the UART interface.
		Timing	Assertion/Negation—An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to sample the data on SIN.

Table 16-2. DUART Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
UART_SOUT[1:2]/ DSP_UART_SOUT	O	Serial data out. The serial data output signals for the UART1, UART2, or DSP_UART are set (mark condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on these signals, with the least significant bit transmitted first.	
		State Meaning	Asserted/Negated—Represents the data transmitted on the respective UART interface.
		Timing	Assertion/Negation—An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to update and drive the data on SOUT.
$\overline{\text{UART\_CTS}}$[1:2]	I	Clear to send. Connected to the respective $\overline{\text{RTS}}$ outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal.	
		State Meaning	Asserted/Negated—Represent the clear to send condition for their respective UART.
		Timing	Assertion/Negation—Sampled at the rising edge of every system clock.
$\overline{\text{UART\_RTS}}$[1:2]	O	Request to send. $\overline{\text{C}}$an be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the $\overline{\text{CTS}}$ input of a transmitter, this signal can be used to control serial data flow.	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation—Updated and driven at the rising edge of every system clock.

16.3 Memory Map/Register Definition

There are two complete sets of DUART registers (one for UART1 and one for UART2). The two UARTs are identical, except that the registers for UART1 are located at offsets 0x0_4500 (local), and the registers for UART2 are located at offsets 0x0_4600 (local). Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for either UART1 or UART2.

The registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to Section 16.3.1.7, “Line Control Registers (ULCR1 and ULCR2),” for more information on ULCR[DLAB].

All DUART registers are one byte wide; reads and writes to these registers must be byte-wide operations. Table 16-3 provides a register summary with references to the section and page that contain detailed information about each register. Undefined byte address spaces within offset 0x4000–0x4FFF are reserved.

Table 16-3. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
0x0_4500	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	16.3.1.1/16-5
	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	16.3.1.2/16-6
	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	16.3.1.3/16-6

Table 16-3. DUART Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x0_4501	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	16.3.1.4/16-8
	UDMB—ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4502	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	16.3.1.5/16-9
	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	16.3.1.6/16-10
	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	16.3.1.12/16-16
0x0_4503	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	16.3.1.7/16-11
0x0_4504	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	0x00	16.3.1.8/16-13
0x0_4505	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	16.3.1.9/16-14
0x0_4506	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	0x00	16.3.1.10/16-15
0x0_4507	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	16.3.1.11/16-16
0x0_4510	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	16.3.1.13/16-17
0x0_4600	URBR—ULCR[DLAB] = 0 UART2 receiver buffer register	R	0x00	16.3.1.1/16-5
	UTHR—ULCR[DLAB] = 0 UART2 transmitter holding register	W	0x00	16.3.1.2/16-6
	UDLB—ULCR[DLAB] = 1 UART2 divisor least significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4601	UIER—ULCR[DLAB] = 0 UART2 interrupt enable register	R/W	0x00	16.3.1.4/16-8
	UDMB—ULCR[DLAB] = 1 UART2 divisor most significant byte register	R/W	0x00	16.3.1.3/16-6
0x0_4602	UIIR—ULCR[DLAB] = 0 UART2 interrupt ID register	R	0x01	16.3.1.5/16-9
	UFCR—ULCR[DLAB] = 0 UART2 FIFO control register	W	0x00	16.3.1.6/16-10
	UAFR—ULCR[DLAB] = 1 UART2 alternate function register	R/W	0x00	16.3.1.12/16-16
0x0_4603	ULCR—ULCR[DLAB] = x UART2 line control register	R/W	0x00	16.3.1.7/16-11
0x0_4604	UMCR—ULCR[DLAB] = x UART2 MODEM control register	R/W	0x00	16.3.1.8/16-13
0x0_4605	ULSR—ULCR[DLAB] = x UART2 line status register	R	0x60	16.3.1.9/16-14
0x0_4606	UMSR—ULCR[DLAB] = x UART2 MODEM status register	R	0x00	16.3.1.10/16-15
0x0_4607	USCR—ULCR[DLAB] = x UART2 scratch register	R/W	0x00	16.3.1.11/16-16
0x0_4610	UDSR—ULCR[DLAB] = x UART2 DMA status register	R	0x01	16.3.1.13/16-17

16.3.1 Register Descriptions

The following sections describe the UART1 and UART2 registers.

16.3.1.1 Receiver Buffer Registers (URBR1 and URBR2)

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, Section 16.3.1.9, “Line Status Registers (ULSR1 and ULSR2).” Figure 16-2 shows the receiver buffer registers. Note that these registers have same offset as the UTHRs.

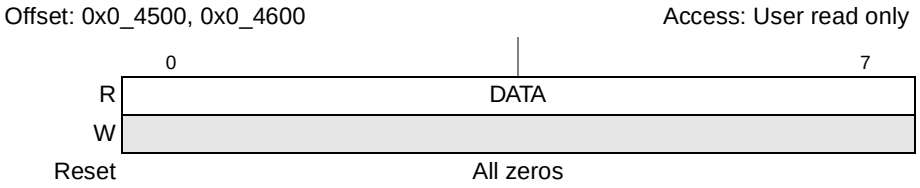


Figure 16-2. Receiver Buffer Registers (URBR1 and URBR2)

Table 16-4 describes URBR.

Table 16-4. URBR Field Descriptions

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus [read only]

16.3.1.2 Transmitter Holding Registers (UTHR1 and UTHR2)

A write to these 8-bit registers causes the UART devices to transfer 5 to 8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR is the first byte onto the bus. UDSR[TXRDY] indicates when the FIFO is full. Refer to Table 16-21 and Table 16-22.

Figure 16-3 shows the bits in the UTHRs.

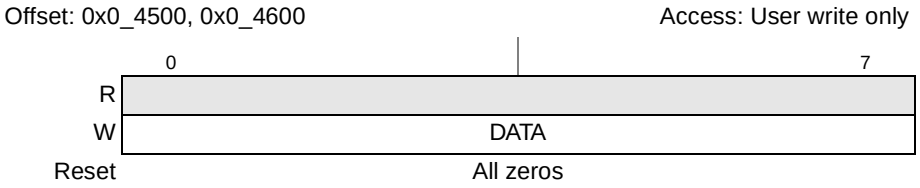


Figure 16-3. Transmitter Holding Registers (UTHR1 and UTHR2)

Table 16-5 describes the UTHR.

Table 16-5. UTHR Field Descriptions

Bits	Name	Description
0–7	DATA	Data that is written to UTHR [Write only]

16.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB)

UDLB is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore, the desired baud rate = platform clock frequency ÷ (16 × [UDMB||UDLB]). Equivalently,

[UDMB||UDLB:0b0000] = platform clock frequency/desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 16-8.

Figure 16-4 shows the bits in the UDMBs.

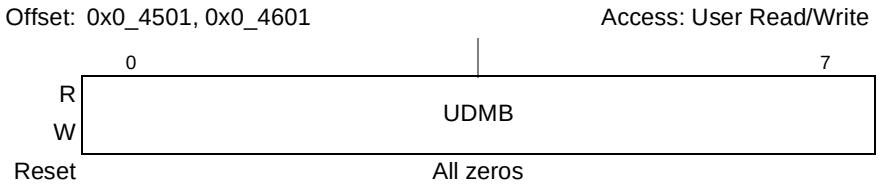


Figure 16-4. Divisor Most Significant Byte Registers (UDMB1 and UDMB2)

Table 16-6 describes the UDMB.

Table 16-6. UDMB Field Descriptions

Bits	Name	Description
0–7	UDMB	Divisor most significant byte

Figure 16-5 shows the bits in the UDLBs.

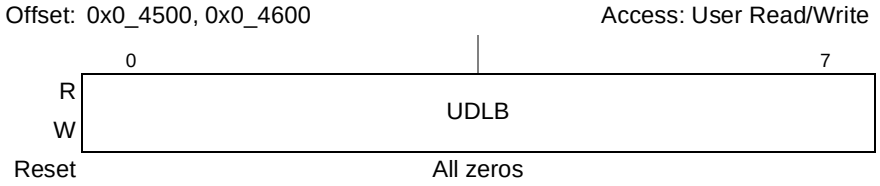


Figure 16-5. Divisor Least Significant Byte Registers (UDLB1 and UDLB2)

Table 16-7 describes the UDLB.

Table 16-7. UDLB Field Descriptions

Bits	Name	Description
0–7	UDLB	Divisor least significant byte. This is concatenated with UDMB.

Table 16-8 shows baud rate for a variety of input clock frequencies.

Table 16-8. Baud Rate Examples

Baud Rate (Decimal)	Divisor		Input Clock (System Clock) Frequency (MHz)	Percent Error (Decimal)
	Decimal	Hex		
9,600	1736	6C8	266	0.0064
19,200	868	364	266	0.0064
38,400	434	1B2	266	0.0064
56,000	298	12A	266	0.1280
128,000	130	82	266	0.1600

Table 16-8. Baud Rate Examples (continued)

Baud Rate (Decimal)	Divisor		Input Clock (System Clock) Frequency (MHz)	Percent Error (Decimal)
	Decimal	Hex		
256,000	65	41	266	0.1600
9,600	2170	87A	333	0.0064
19,200	1085	43D	333	0.0064
38,400	543	21F	333	0.0858
56,000	372	174	333	0.0064
128,000	163	A3	333	0.1472
256,000	81	51	333	0.4672

To get the percent error value, the following three steps are taken:

1. The input clock frequency (ICF) is divided by the actual frequency input (AFI) to get the correct divisor value (ICF/AFI, where AFI = baud rate x 16).
2. The divisor value is subtracted from 1.
3. The result from the step two is multiplied by 100 to calculate the final percent error. The result is calculated in absolute value (no negative numbers).

   These steps can be described with the following equation:

   Percent error value = $(1 - AFI/ICF) \times 100$

16.3.1.4 Interrupt Enable Registers (UIER1 and UIER2)

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Figure 16-6 shows the bits in the UIER.

Offset: 0x0_4501, 0x0_4601 Access: User Read/Write

	0–3	4	5	6	7
R W	—	EMSI	ERLSI	ETHREI	ERDAI
Reset	All zeros				

Figure 16-6. Interrupt Enable Registers (UIER1 and UIER2)

Table 16-9 describes the UIER fields.

Table 16-9. UIER Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4	EMSI	Enable MODEM status interrupt 0 Mask interrupts caused by UMSR[DCTS] being set. 1 Enable and assert interrupts when UMSR[CTS] changes state.
5	ERLSI	Enable receiver line status interrupt 0 Mask interrupts when ULSR's overrun, parity error, framing error, or break interrupt bits are set. 1 Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set.
6	ETHREI	Enable transmitter holding register empty interrupt 0 Mask interrupt when ULSR[THRE] is set. 1 Enable and assert interrupts when ULSR[THRE] is set.
7	ERDAI	Enable received data available interrupt 0 Mask interrupt when new receive data is available or receive data time-out has occurred. 1 Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in FIFO mode.

16.3.1.5 Interrupt ID Registers (UIIR1 and UIIR2)

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are as follows:

1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty
4. MODEM status

See Table 16-11 for more details.

When the UIIR is read, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated DUART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

Figure 16-7 shows the bits in the UIIR.

Offset: 0x0_4502, 0x0_4602 Access: User 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 16-7. Interrupt ID Registers (UIIR1 and UIIR2)

Table 16-10 describes the fields of the UIIR.

Table 16-10. UIIR Field Descriptions

Bits	Name	Description
0–1	FE	FIFOs enabled. Reflects the setting of UFCR[FEN].
2–3	—	Reserved
4	IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 16-11. IID3 is set along with IID2 only when a time out interrupt is pending for FIFO mode.
5–6	IID2–IID1	Interrupt ID bits identify the highest priority pending interrupt as indicated in Table 16-11.
7	IID0	IID0 indicates when an interrupt is pending. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

The bits contained in the UIIR registers are described in Table 16-11.

Table 16-11. UIIR IID Bits Summary

IID3–IID0	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0001	—	—	—	—
0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Reading the line status register
0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode.	Reading the receiver buffer register or if the number of bytes in the receiver FIFO drops below the trigger level.
1100	Second	Character time-out	No characters were removed from or input to the receiver FIFO during the last four character times and at least one character is in the receiver FIFO.	Reading the receiver buffer register
0010	Third	UTHR empty	Transmitter holding register is empty.	Reading UIIR or writing to UTHR
0000	Fourth	MODEM status	$\overline{\text{CTS}}$ input value changed since last read of UMSR.	Reading UMSR

16.3.1.6 FIFO Control Registers (UFCR1 and UFCR2)

UFCR is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

UFCR bits cannot be programmed unless FIFO enable bits are set. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all of the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self clearing.

Figure 16-8 shows the bits in the UFCRs.

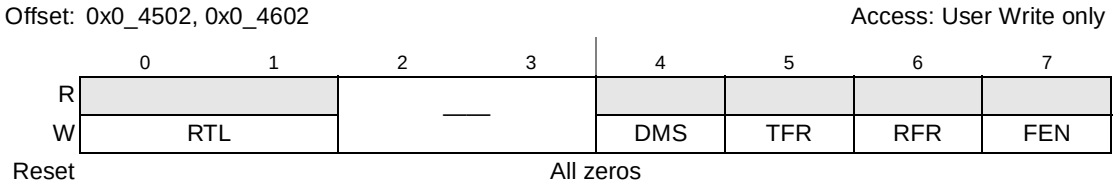


Figure 16-8. FIFO Control Registers (UFCR1 and UFCR2)

Table 16-12 describes the fields of the UFCRs.

Table 16-12. UFCR Field Descriptions

Bits	Name	Description
0–1	RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals RTL value. 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3	—	Reserved
4	DMS	DMA mode select. See Section 16.4.5.2, “DMA Mode Select” 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5	TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6	RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7	FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Transmitter and receiver FIFOs are enabled.

16.3.1.7 Line Control Registers (ULCR1 and ULCR2)

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing ULCR, the software should not rewrite the ULCR while valid transfers on the UART bus are active. The software should not rewrite the ULCR until the last STOP bit is received and no new characters are being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See Table 16-14. ULCR[NSTB] defines the number of STOP bits to be sent at the end of the data transfer. The receiver checks only the first STOP bit, regardless of the number

of STOP bits selected. The word length select bits (1 and 0) define the number of data bits transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Figure 16-9 shows the bits in the ULCRs.

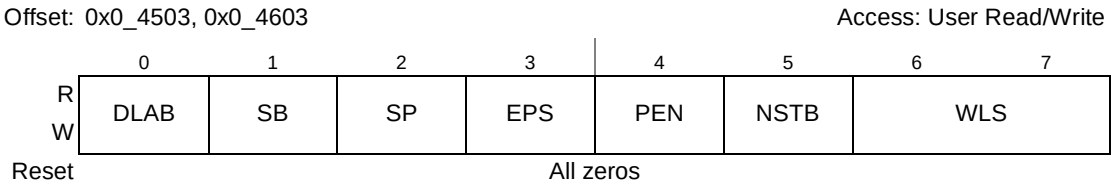


Figure 16-9. Line Control Register (ULCR1 and ULCR2)

Table 16-13 describes the ULCR fields.

Table 16-13. 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 UDMB, UDLB, and UAFR.
1	SB	Set break 0 Send normal UTHR data onto the SOUT signal. 1 Force logic 0 to be on SOUT. Data in the UTHR is not affected.
2	SP	Stick parity 0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected; if PEN = 1 and EPS = 0, mark parity is selected.
3	EPS	Even parity select. See Table 16-14. 0 If PEN = 1 and SP = 0 then odd parity is selected. 1 If PEN = 1 and SP = 0 then even parity is selected.
4	PEN	Parity enable 0 No parity generation and checking. 1 Generate parity bit as a transmitter, and check parity as a receiver.
5	NTSB	Number of STOP bits 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1 1/2 STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6–7	WLS	Word length select. Number of bits that comprise the character length. 00 5 bits 01 6 bits 10 7 bits 11 8 bits

Table 16-14. 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

16.3.1.8 MODEM Control Registers (UMCR1 and UMCR2)

The UMCRs, shown in Figure 16-10, control the interface with the external peripheral device on the UART bus.

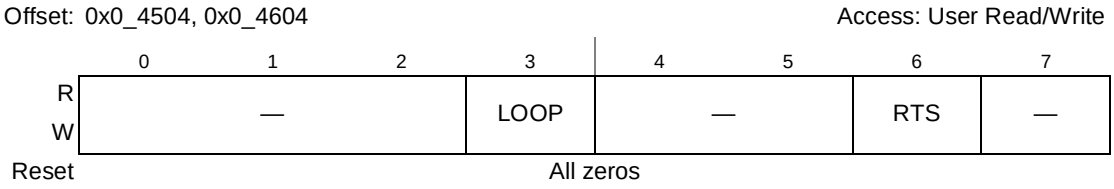


Figure 16-10. Modem Control Register (UMCR1 and UMCR2)

Table 16-15 describes the UMCR fields.

Table 16-15. UMCR Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared
3	LOOP	Local loopback mode 0 Normal operation. 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].
4–5	—	Reserved
6	RTS	Ready to send 0 Negates corresponding $\overline{\text{UART\_RTS}}$ output. 1 Assert corresponding $\overline{\text{UART\_RTS}}$ output. Informs external MODEM or peripheral that the UART is ready for sending/receiving data.
7	—	Reserved

16.3.1.9 Line Status Registers (ULSR1 and ULSR2)

The ULSRs, shown in Figure 16-11, 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.

Offset: 0x0_4505, 0x0_4605 Access: User 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 16-11. Line Status Register (ULSR1 and ULSR2)

Table 16-16 describes the ULSR fields.

Table 16-16. ULSR Field Descriptions

Bits	Name	Description
0	RFE	Receiver FIFO error. 0 Cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt).
1	TEMT	Transmitter empty 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2	THRE	Transmitter holding register empty 0 UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3	BI	Break interrupt 0 Cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
4	FE	Framing error 0 Cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, FE is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then will receive the following new data.
5	PE	Parity error 0 Cleared when ULSR is read or when a new character is loaded into URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.

Table 16-16. ULSR Field Descriptions (continued)

Bits	Name	Description
6	OE	Overrun error 0 Cleared when ULSR is read 1 Before URBR was read, it was overwritten with a new character. The old character is lost. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7	DR	Data ready 0 Cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character was received in the URBR or the receiver FIFO.

16.3.1.10 MODEM Status Registers (UMSR1 and UMSR2)

The UMSRs, shown in Figure 16-12, track the status of the MODEM (or external peripheral device) $\overline{\text{CTS}}$, set for the corresponding UART.

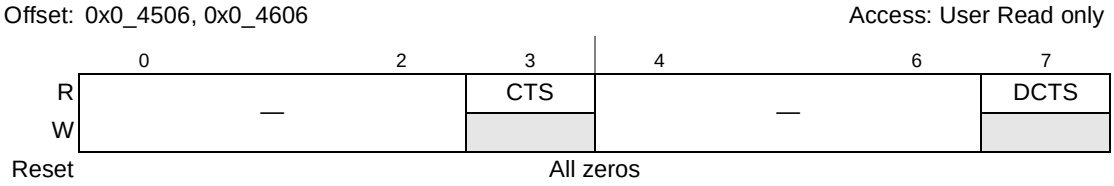


Figure 16-12. Modem Status Register (UMSR1 and UMSR2)

Table 16-17 describes UMSR fields.

Table 16-17. UMSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared
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, should be cleared
7	DCTS	Delta clear to send 0 No change on the corresponding $\overline{\text{CTS}}n$ signal since the last read of UMSR[CTS]. 1 $\overline{\text{CTS}}n$ changed since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition.

16.3.1.11 Scratch Registers (USCR1 and USCR2)

USCR, shown in Figure 16-13, are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

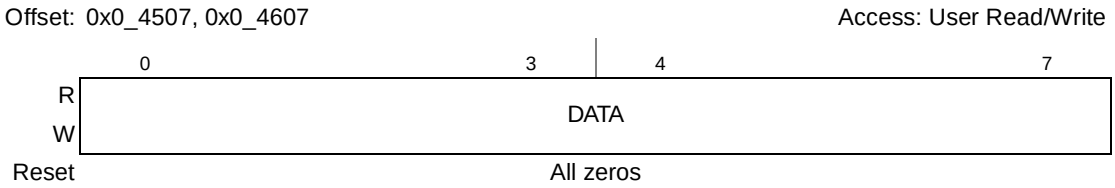


Figure 16-13. Scratch Register (USCR)

Table 16-18 describes USCR fields.

Table 16-18. USCR Field Descriptions

Bits	Name	Description
0–7	DATA	Data

16.3.1.12 Alternate Function Registers (UAFR1 and UAFR2)

The UAFRs, shown in Figure 16-14, allow software to write to both UART1 and UART2 registers simultaneously with the same write operation. The UAFRs also provide a means for the device's performance monitor to track the baud clock.

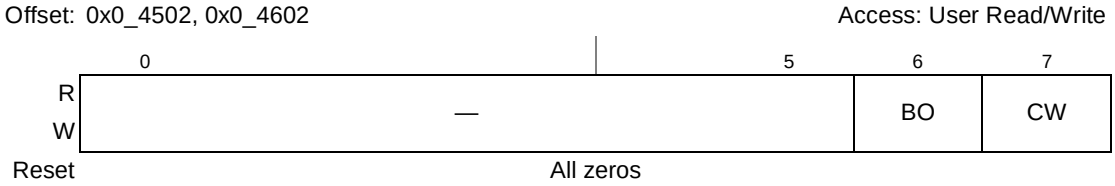


Figure 16-14. Alternate Function Register (UAFR)

Table 16-19 describes UAFR fields.

Table 16-19. UAFR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	BO	Baud clock select 0 The baud clock is not gated off. 1 The baud clock is gated off.
7	CW	Concurrent write enable 0 Disables writing to both UART1 and UART2. 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART1 is also a write to the corresponding register in UART2 and vice versa.

16.3.1.13 DMA Status Registers (UDSR1 and UDSR2)

The DMA status registers (UDSRs), shown in Figure 16-15, return transmitter and receiver FIFO status and provide the ability to assist DMA data operations to and from the FIFOs.

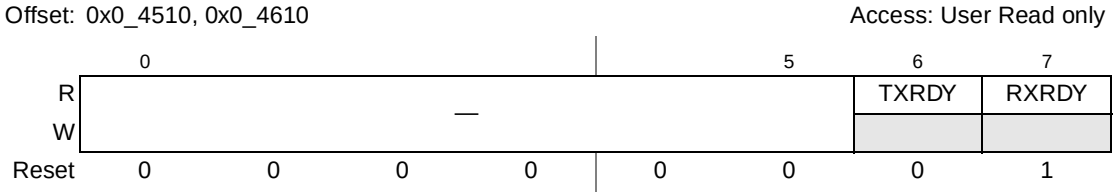


Figure 16-15. DMA Status Register (UDSR)

Table 16-20 describes the fields of the UDSRs.

Table 16-20. UDSR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	TXRDY	Transmitter ready. Reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by UFCR[DMS] and UFCR [FEN]. 0 The bit is cleared, as shown inTable 16-22. 1 This bit is set, as shown in Table 16-21.
7	RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by UFCR[DMS] and UFCR [FEN]. 0 The bit is cleared, as shown in Table 16-24. 1 This bit is set, as shown in Table 16-23.

Table 16-21. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 16-22. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear while the transmitter FIFO is not yet full.

Table 16-23. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 16-24. UDSR[RXRDY] Cleared

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

16.4 Functional Description

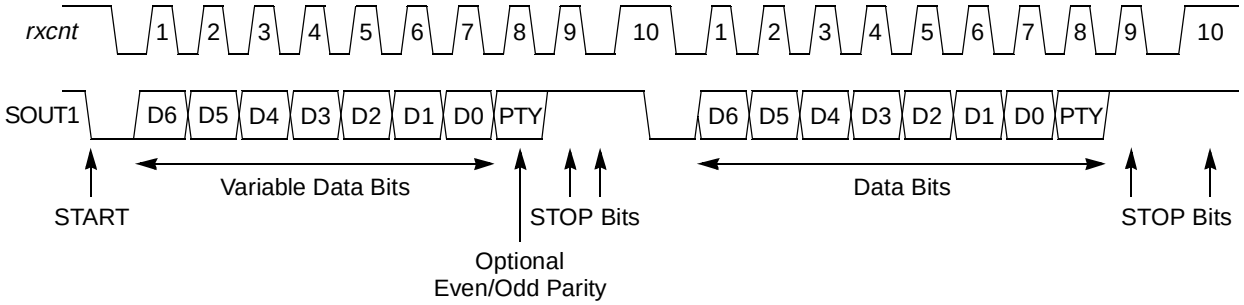
The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the system clock signal.

The transmitter accepts parallel data with a write access to UTHR. In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see Section 16.4.5, “FIFO Mode.” The transmitting registers convert the data to a serial bit stream by inserting the appropriate START, STOP, and optional parity bits. Finally, the registers output a composite serial data stream on the channel transmitter serial data output (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data on the channel receiver serial data input (SIN), converts the data into parallel format, and checks for START, STOP, and parity bits. In FIFO mode, the receiver removes the START, STOP, and parity bits and then transfers the assembled character from the receiver buffer, or receiver FIFO. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt-driven.

16.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 16-16. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.



Two 7-Bit Data Transmissions with Parity and 2-Bit STOP Transactions

Figure 16-16. UART Bus Interface Transaction Protocol Example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer (least significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loopback mode, different errors, and FIFO mode.

16.4.1.1 START Bit

A write to UTHR generates a START bit on the SOUT signal. Figure 16-16 shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in ULCR.When the bus is idle, SOUT is high.

16.4.1.2 Data Transfer

Each data transfer contains 5, 6, 7, or 8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time, a START bit is generated followed by 5 to 8 of the data bits previously written to the UTHR. The data bits are driven from the least- to the most-significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to UTHR.

16.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see Section 16.3.1.7, “Line Control Registers (ULCR1 and ULCR2).” Both the receiver and transmitter parity definitions must agree before

transferring data. When receiving data, a parity error can occur if an unexpected parity value is detected (see Section 16.3.1.9, “Line Status Registers (ULSR1 and ULSR2)”).

16.4.1.4 STOP Bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The user can program the length of the STOP bit(s) in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

16.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the system clock input and dividing the input by any divisor from 1 to $2^{16} - 1$.

5. The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

   Baud rate = (1/16) × (system clock frequency/divisor value)

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

1. The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling UAFR[BO]. This can be used to determine baud-rate errors.

16.4.3 Local Loopback Mode

Local loopback mode is provided for diagnostic testing. The data written to UTHR can be read from the receiver buffer register (URBR) of the same UART. In this mode, the MODEM control register UMCR[RTS] is internally tied to the MODEM status register UMSR[CTS]. The transmitter SOUT is set to a logic 1 and the receiver SIN is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The $\overline{\text{CTS}}$ (input signal) is disconnected, $\overline{\text{RTS}}$ is internally connected to $\overline{\text{CTS}}$, and the $\overline{\text{RTS}}$ (output signal) becomes inactive. In this diagnostic mode, data that is transmitted is immediately received. In local loopback mode the transmit and receive data paths of the DUART can be verified. Note that in local loopback mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

16.4.4 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus. Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

16.4.4.1 Framing Error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set. Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

16.4.4.2 Parity Error

When unexpected parity values are encountered while receiving data, a parity error occurs and ULSR[PE] is set. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

16.4.4.3 Overrun Error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set. In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

16.4.5 FIFO Mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (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. UDSR[TXRDY] indicate when the transmitter FIFO is full. When in FIFO mode, data written to UTHR is placed into the transmitter FIFO. The first byte written to UTHR is the first byte onto the UART bus.

16.4.5.1 FIFO Interrupts

In FIFO mode, the UIER[ERDAI] is set when a time-out interrupt occurs. A receive data time-out generates a maskable interrupt condition (through UIER[ERDAI]). See Section 16.3.1.4, “Interrupt Enable Registers (UIER1 and UIER2)”.

UIIR indicates whether the FIFOs are enabled. UIIR[IID3] is set only for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and at least one character is in the receiver FIFO. The character time-out interrupt (controlled by UIIR[IID]) is cleared when URBR is read. See Section 16.3.1.5, “Interrupt ID Registers (UIIR1 and UIIR2).”

UIIR[FE] indicates whether FIFO mode is enabled.

16.4.5.2 DMA Mode Select

UDSR[RXRDY] reflects the status of the receiver FIFO or URBR. In mode 0 (UFCR[DMS] is cleared), UDSR[RXRDY] is cleared when at least one character is in the receiver FIFO or URBR; it is set when there are no more characters in the receiver FIFO or URBR. This occurs regardless of the UFCR[FEN] setting. 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; it is set when there are no more characters in the receiver FIFO.

UDSR[TXRDY] reflects the status of the transmitter FIFO or UTHR. In mode 0 (UFCR[DMS] is cleared), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR; it is set after the first character is loaded into the transmitter FIFO or UTHR. This occurs regardless of the UFCR[FEN] setting. 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; it is set when the transmitter FIFO is full.

See Section 16.3.1.13, “DMA Status Registers (UDSR1 and UDSR2),” for a complete description of the USDR[RXRDY] and USDR[TXRDY] bits.

16.4.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 0 (UIIR[0]), is cleared. UIER is used to mask specific interrupt types. See Section 16.3.1.4, “Interrupt Enable Registers (UIER1 and UIER2).”

When the interrupts are disabled in UIER, polling software can not use UIIR[0] to determine whether the UART is ready for service. Software must monitor the appropriate ULSR and UMSR bits. UIIR[0] can be used for polling if the interrupts are enabled in UIER.

16.5 DUART Initialization/Application Information

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited and guarded area. (That is, the WIMG setting in the MMU needs to be 0b01x1.)
- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-length operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.

3. Set the data attributes and control bits of the external MODEM or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.
6. Poll UIIR if the interrupts generated by the DUART are masked.

Chapter 17
JTAG/Testing Support

17.1 Overview

The device provides a JTAG (Joint Test Action Group) interface to facilitate boundary-scan testing. The JTAG interface complies to the IEEE 1149.1 boundary-scan specification. For additional information about JTAG operations, refer to the IEEE 1149.1 specification.

The JTAG interface consists of a set of five signals, three JTAG registers (see Section 17.3, “JTAG Registers and Scan Chains,”) and a test access port (TAP) controller, described in the following sections. A block diagram of the JTAG interface is shown in Figure 17-1.

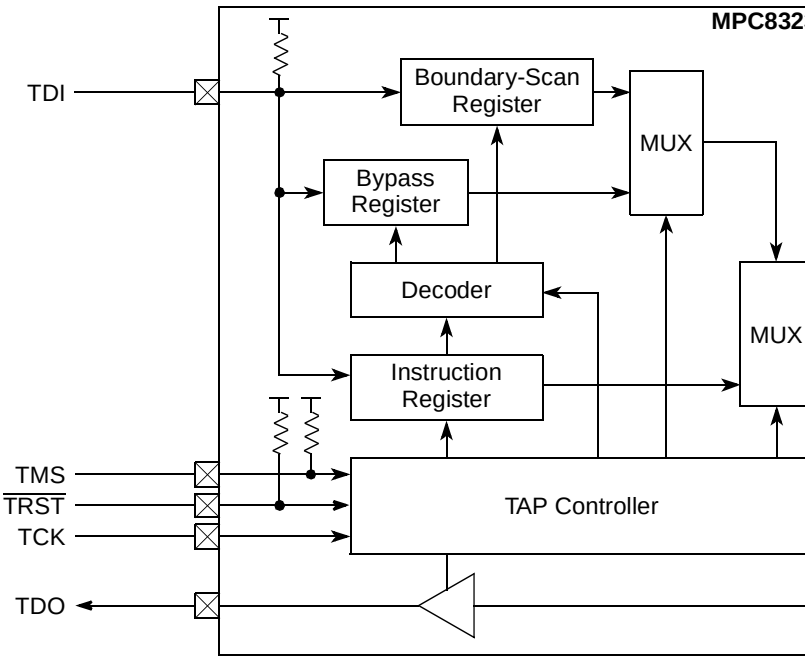


Figure 17-1. JTAG Interface Block Diagram

17.2 JTAG Signals

The device provides the following five dedicated JTAG signals:

- Test data input (TDI)
- Test data output (TDO)
- Test mode select (TMS)

- Test reset ($\overline{\text{TRST}}$)
- Test clock (TCK)

The TDI and TDO signals input and output all instructions and data to the JTAG scan registers. JTAG operations are controlled by the TAP controller through the TMS and TCK signals. Boundary-scan data is latched by the TAP controller on the rising edge of the TCK signal. The $\overline{\text{TRST}}$ signal is specified as optional by the IEEE 1149.1 specification, and is used to reset the TAP controller asynchronously. The assertion of the $\overline{\text{TRST}}$ signal at power-on reset ensures that the JTAG logic does not interfere with the normal operation of the device.

17.2.1 External Signal Descriptions

The JTAG signals are summarized in Table 17-1.

Table 17-1. JTAG Test Signals Summary

Name	Description	Functional Block	Function	Reset Value	I/O
TCK	Test clock	Debug	Clock for JTAG testing.	—	I
TDI	Test data input		Serial input for instructions and data to the JTAG test subsystem. Internally pulled up.	—	I
TDO	Test data output		Serial data output for the JTAG test subsystem. High impedance except when scanning out data.	High impedance	O
TMS	Test mode select		Carries commands to the TAP controller for boundary scan operations. Internally pulled up.	—	I
$\overline{\text{TRST}}$	Test reset		Resets the TAP controller asynchronously. Internally pulled up.	—	I

Table 17-2 shows detailed descriptions of the JTAG test signals.

Table 17-2. JTAG Test—Detailed Signal Descriptions

Signal	I/O	Description	
TCK	I	JTAG test clock.	
		State Meaning	Asserted/Negated—Should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the TAP are clocked in on the rising edge. Changes to the TAP output signals occur on the falling edge. The test logic allows TCK to be stopped.
		Timing	See IEEE 1149.1 specification for more details.
TDI	I	JTAG test data input.	
		State Meaning	Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 specification for more details.

Table 17-2. JTAG Test—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TDO	O	JTAG test data output.	
		State Meaning	Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		Timing	See IEEE 1149.1 specification for more details.
TMS	I	JTAG test mode select.	
		State Meaning	Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 specification for more details.
$\overline{\text{TRST}}$	I	JTAG test reset.	
		State Meaning	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the device. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor. Negated— Normal operation.
		Timing	See IEEE 1149.1 specification for more details.

17.3 JTAG Registers and Scan Chains

The bypass, boundary-scan, and instruction JTAG registers and their associated scan chains are mandatory for compliance with the IEEE 1149.1 specification.

- Bypass register. The bypass register is a single-stage register used to bypass the boundary-scan latches of the device during board-level boundary-scan operations involving components other than the device. The use of the bypass register reduces the total scan string size of the boundary-scan test.
- Boundary-scan registers. The JTAG interface provides a chain of registers dedicated to boundary-scan operations. To be JTAG-compliant, these registers cannot be shared with any functional registers of the device. The boundary-scan register chain includes registers controlling the direction of the input/output drivers, in addition to the registers reflecting the signal value received or driven.

  The boundary-scan registers capture the input or output state of the device's signals during a Capture_DR TAP controller state. When a data scan is initiated following the Capture_DR state, the sampled values are shifted out through the TDO output while new boundary-scan register values are shifted in through the TDI input. At the end of the data scan operation, the boundary-scan registers are updated with the new values during an update_DR TAP controller state.
- Instruction register. The 8-bit JTAG instruction register serves as an instruction and status register. As TAP controller instructions are scanned in through the TDI input, the TAP controller status bits are scanned out through the TDO output.

- TAP controller. The device provides a standard JTAG TAP controller that controls instruction and data scan operations. The TMS signal controls the state transitions of the TAP controller.

Chapter 18
System Interface

The QUICC Engine system interface consists of functions that interface to the system bus and core processor. The system interface includes the following:

- Serial DMA
- Interrupt Controller

The first interface to the system is the serial DMA. This Serial DMA (SDMA) is responsible for transferring data from the QUICC Engine module to memory. The SDMA supports all the communication channels in the QUICC Engine module, and automatically transfers data to/from the correct queue. The SDMA is also used to transfer parameters to/from external memory. The SDMA supports two buses.

The second interface to the system is the interrupt controller. This controller is accessed by the core processor to verify the cause of an interrupt from the QUICC Engine module.

18.1 Serial DMA

The QUICC Engine module has two physical serial DMA (SDMA) channels. One channel interfaces with coherent system bus, the other channel interfaces with the secondary bus.

The QUICC Engine module has two physical serial DMA (SDMA) channels. The QUICC Engine module implements two dedicated virtual SDMA channels for each peripheral (for example, UCC, SPI), one for the receiver and one for the transmitter. Additional virtual channels are available for general purpose accesses to external memory.

18.1.1 Data Paths

Figure 18-1 is a simplified block diagram that shows the data paths. They may be configured with one DDR bus (32 bit data). As shown in the figures, depending on the external bus configuration, the secondary bus of the QUICC Engine module is connected to the Local bus. Accesses of the QUICC Engine module to the secondary bus are not seen on the coherent system bus. This allows for concurrent bus transactions on the two buses (marked as ‘1’ and ‘2’ in the figure).

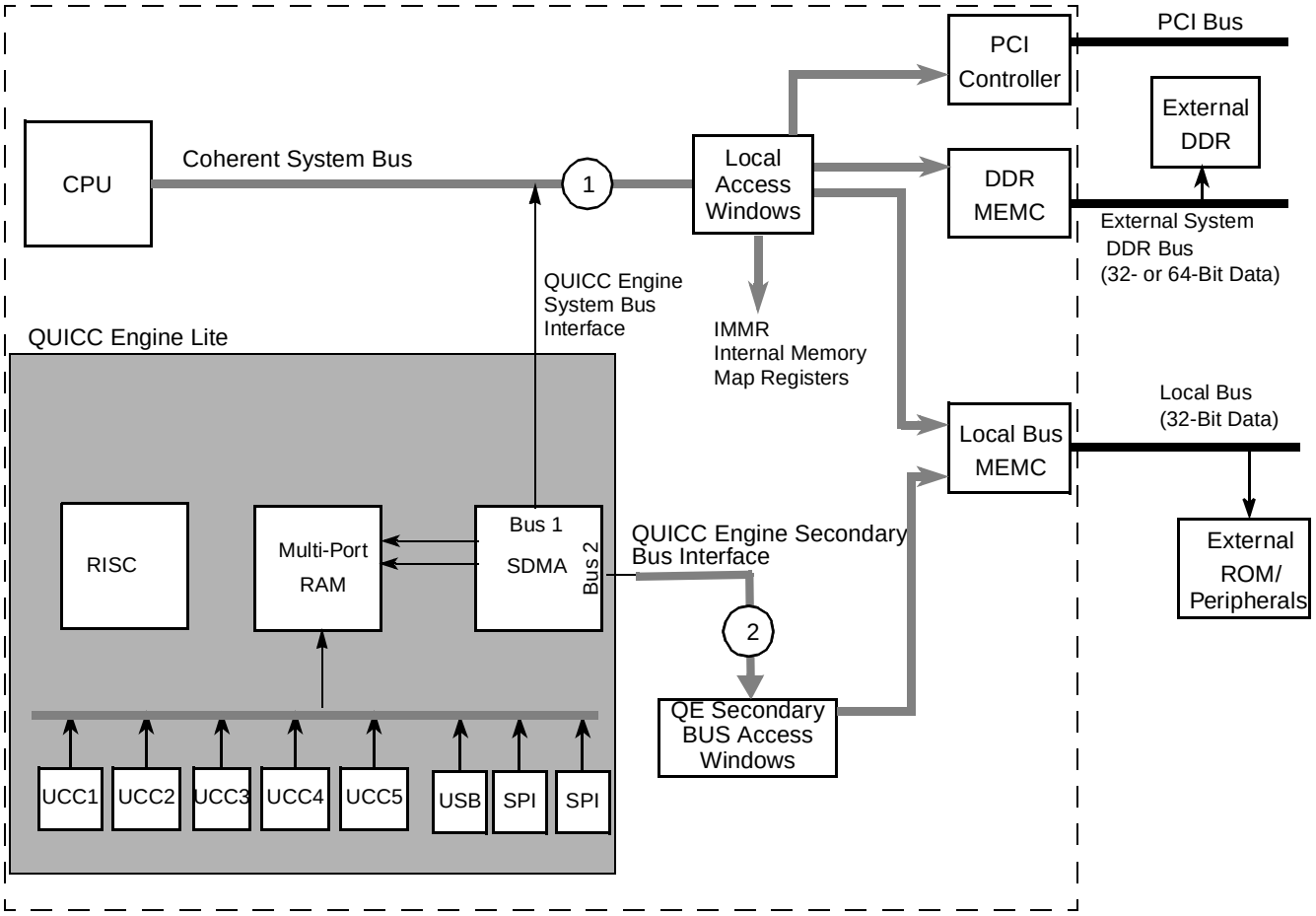


Figure 18-1. Data Paths for MPC8323E

The ‘Local Access Windows’ block shown in the figure, is responsible for distribution of coherent system bus transactions to the various possible targets (for example, PCI, DDR memory controller) based on the transaction’s address.See Section 5.2, “Local Memory Map Overview and Example” for further details. Similarly the “QUICC Engine Secondary Bus Access Windows” block is responsible for distribution of QUICC Engine Secondary Bus transactions to the Local Bus memory controller The SDMA channel must be configured for big-endian byte ordering for accessing buffer data. The big-endian byte ordering format is programmed in the receive and transmit bus mode registers associated with the peripherals (UCC, SPI). Refer to each protocol of these peripherals for programming of this feature.

18.1.2 SDMA and Bus Error

If a bus error occurs on an SDMA access from the QUICC Engine module, the following occurs:

1. The QUICC Engine module generates a unique maskable interrupt (SDMR[BER_1_MSK] and SDMR[BER_2_MSK]) in the SDMA status register (SDSR). See Section 18.1.8.2, “Serial DMA Mode Register (SDMR).” and Section 18.1.8.1, “Serial DMA Status Register (SDSR).”

2. The core processor in its interrupt service routine reads the SDSR to determine which bus generated the error (System bus or secondary bus),
3. System recovery from a bus error is dependent upon how the user configures SDMR[SBER_1] and SDMR[SBER_2]. See Section 18.1.8.2, “Serial DMA Mode Register (SDMR).” There are two modes:
   a) The QUICC Engine module disables the peripheral or thread that is associated with the bus error and continues to operate as usual on all other peripherals (this is the default). The recovery sequence from an error in this mode is based on re-initialization of the peripheral(s) associated with this error,

      - or -

   b) The QUICC Engine module stops all activity, and the chip must be reset by asserting the soft reset signal, $\overline{\text{SRESET}}$, (reset command to the QUICC Engine Command Register is not sufficient).

In general, it should be noted that the core processor, depending on how the user programmed it, may read the SDMA address register (SDTA1 or SDTA2) to determine the address the bus error occurred on, and the SDMA SNUM register (SDTM1 or SDTM2) to determine which peripheral or thread was being serviced by the SDMA virtual channel. See Section 19.7, “Serial Number (SNUM),” for an explanation of the term SNUM.

The SDTA1 and the SDTM1 registers store information related to accesses to the system bus; SDTA2 and SDTM2 store information related to accesses to the secondary bus. These four registers are not updated with the address and SNUM of subsequent transactions as long as their respective event bit (in SDSR register) is set.

See Section 18.1.8.5, “Serial DMA Transfer Address Registers (SDTA1 and SDTA2)” and Section 18.1.8.6, “Serial DMA Transfer Communication Channel Number Registers (SDTM1 and SDTM2).”

18.1.2.1 Simple Recovery from Bus Error

The simplest recovery from a bus error is an QUICC Engine reset (through QUICC Engine Command Register) followed by an overall QUICC Engine module re-initialization procedure, regardless of the peripheral that has actually caused the error. The reasoning is that for some applications stopping one peripheral lead to a chain reaction that ultimately disrupt the correct interworking operation of the QUICC Engine module. For debug purpose it is valuable that the QUICC Engine module continue to operate but a selective recovery does not provide any added value. This non-distinctive procedure also reduces the complexity of the recovery flow.

18.1.2.2 Selective Peripheral Recovery Procedure

Selective recovery can be a complex procedure due to the following:

- A peripheral (UCC) can have dependencies with other peripherals (as a part of an interworking controller like ATM switch etc.), and stopping it would lead to an erratic behavior of the entire controller.

- The ATM and UEC are multi-thread controllers and SNUM to peripheral/controller translation requires the user to maintain association tables.
- Status for multiple bus errors is not maintained, so in theory there might be additional non-reported bus errors and the recovery will not be complete.

It is possible to perform a selective single peripheral re-initialization procedure only under the following conditions:

- The SNUM is of a peripheral which implement one of the following termination protocols:
  - HDLC
  - Transparent
  - UART
  - BiSync
  - QMC
  - SPI
  - USB

In other cases it is recommended to reset the QUICC Engine module and do a full initialization sequence.

18.1.3 SDMA and Reset

When the QUICC Engine is reset, through CECR[RST] (See Section 19.3.1, “QUICC Engine Command Register (CECR)”) the SDMA continues to process outstanding transactions in its FIFOs although the data related to these transactions may be corrupted. During system reset ($\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$) all SDMA FIFOs are flushed and all outstanding transactions are stopped.

18.1.4 Bus Arbitration

This section explains the bus arbitration in the QUICC Engine block over the system bus, secondary bus, and Multi-user RAM bus.

18.1.4.1 Arbitration Over the System Bus

On the MPC8323E, the QUICC Engine block arbitrates over the system bus by requesting access to the bus from the system arbiter. The arbiter supports four levels of priorities. The QUICC Engine module also asserts a REPEAT signal to the arbiter for transactions which are longer than one burst (32 bytes) on the System Bus. In this way, the arbiter can optimize bus grants to allow for Page Hits on the DRAMs, by allowing back to back cycles to subsequent addresses.

The SDMA requests the bus from the arbiter at two possible priority levels. When the SDMA is in normal state it requests access at a priority level which is programmable by the user in SDMR[EB1_PR] bit field. When the SDMA is in emergency state it requests access at the highest priority level which the arbiter supports. Registers SDTR1 and SDHY1 program the threshold and hysteresis values which affect the conditions for SDMA normal and emergency states. Note also that it is possible to globally mask the emergency state priority requests in SDMR[EB1_MSK]—thus enforcing the priority set in SDMR[EB1_PR] regardless of SDMA state.

The SDMA will assert the highest priority request (emergency state) for any one of the following reasons:

- When one of the FIFOs in the QUICC Engine module reaches an emergency state (too full on Rx or too empty in the middle of a frame during Tx)
- When the internal SDMA data buffers are filled beyond a certain level (which is programmable in SDTR/SDHY registers).
- When the internal SDMA command queue is filled beyond a certain level (which is programmable in SDTR/SDHY registers).

18.1.4.2 Arbitration Over the Secondary Bus

On the MPC8323E, arbitration over the Secondary bus is not affected by internal state of the SDMA. Generally arbitration occurs in two places: Local Bus Memory Controller and Secondary DDR SDRAM Memory Controller. In both places arbitration is between the QUICC Engine module, which accesses the resource through its secondary bus, and another master which initiated a transaction on the System Bus. In the case of the Local Bus Memory Controller, the QUICC Engine module has the lower priority. In the case of the Secondary DDR SDRAM Controller arbitration is based on rotating priority.

18.1.4.3 Arbitration Over the Multi-User RAM Bus

Priority request to the multi-user RAM is also asserted by the SDMA if needed for similar reasons as explained above. It is possible to globally mask the high priority request in SDMR[ER1_MSK] and SDMR[ER2_MSK].

18.1.5 SDMA and Snooping

The SDMA in the QUICC Engine module is able to dynamically signal whether or not the current cycle on the system bus is to be snooped for cache coherency. This is done by asserting the GBL signal to the core processor. Refer to each protocol of these peripherals for programming of this feature, for example, for Ethernet, refer to Section 29.5.4, “Bus Mode Register (BMRx).”

The user may globally disable the snooping by resetting the SDMR[GLB_1_MSK] bit in the SDMA. Note that the secondary bus of the QUICC Engine module cannot be snooped by the core processor.See Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”

18.1.6 Bus Selection Mechanisms

For each transaction initiated by the RISC, the SDMA in the QUICC Engine module must select the destination bus: Either the System bus or the secondary bus. The SDMA implements two mechanisms for bus selection:

- User programmable window. In this way, the user programs the SDAQR and SDAQMR registers in the SDMA with the base address of the external memory window to be mapped to the System bus and the size of the window. Any access matching this window is automatically routed to the System bus. Other accesses are routed to the secondary bus. The minimum window size is 64K bytes. It is possible to program the base address of the secondary bus instead of the system bus.

- User programmable bus selection by protocol. In this mode (which is compatible with the MPC82xx/85xx family of devices) the SDMA selects the bus according to the bit programmed in each protocol. Refer to each protocol of these peripherals for programming of this feature, for example, for Ethernet, refer to Section 29.5.4, “Bus Mode Register (BMRx).”

The selection between these two modes is made in SDMR[ADR_SEL].

18.1.7 SDMA Internal Resource

The SDMA requires temporary buffering for some data in the multi-user RAM. The base address for the SDMA temporary buffer is programmed in the SDEBCR register. The base address must be aligned to a 4Kbyte boundary. The size of the multi-user RAM needed for this buffering is programmable via the CEN bits field in the SDMR register. The size of the temporary buffer that must be allocated ranges from a minimum of 512 bytes to maximum of 3 Kbytes in the multi-user RAM. Refer for the detailed CEN bits field description for suggested values.

18.1.8 Programming Model of the Serial DMA

All of the DMA registers are accessible via software (through the slave) for read/write.

During reset, all the DMA registers are reset, except SDSR, SDTAx and SDTMx. SDSR bits are cleared by writing ones to them (writing zeros has no effect). After each initialization of the system the bits of SDSR must be cleared before unmasking the interrupt controller mask bits. SDTAx and SDTMx are read-only registers.

18.1.8.1 Serial DMA Status Register (SDSR)

SDSR, shown in Figure 18-2, reports bus error events recognized by the Serial DMA controller on all of the Serial DMA channels. On recognition of a bus error event, the Serial DMA sets the corresponding bit in SDSR. SDSR is a memory mapped register, that can be read at any time. Bits are cleared by writing ones to them. Writing zeros has no effect.

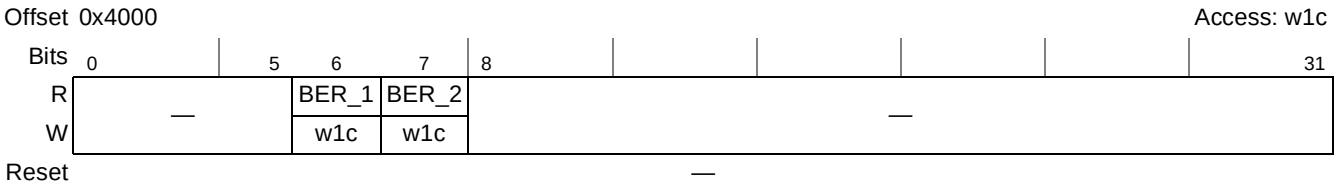


Figure 18-2. Serial DMA Status Register (SDSR)

Table 18-1 describes the SDSR fields.

Table 18-1. SDSR Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	BER_1	Bus 1 error event. Used to indicate that the Serial DMA channel on bus 1 was terminated with an error during a read or write transaction. This bit is cleared by writing ‘1’. Writing ‘0’ has no effect. The Serial DMA bus 1 error address is read from the SDTA1 register. The communication channel number is read from the SDTM1 register. 0 No bus error event occurred on bus 1 1 A bus error event occurred on bus 1
7	BER_2	Bus 2 error event. Used to indicate that the Serial DMA channel on bus 2 terminated with an error during a read or write transaction. This bit is cleared by writing ‘1’. Writing ‘0’ has no effect. The Serial DMA bus 2 error address is read from the SDTA2 register. The communication channel number is read from the SDTM2 register. 0 No bus error event occurred on bus 2 1 A bus error event occurred on bus 2
8–31	—	Reserved, should be cleared.

18.1.8.2 Serial DMA Mode Register (SDMR)

SDMR, shown in Figure 18-3, enables the user to mask the Serial DMA interrupts and emergency mode and to set them manually. For bits 6,7 if an SDMR bit is set, the corresponding interrupt in SDSR is enabled. If the bit is cleared, the corresponding interrupt in SDSR is masked.

Offset 0x4004 Access: Read/Write

Bits	0	1	2	3	5	6	7
R W	GLB_1_MSK	—	ADR_SEL	—		BER_1_MSK	BER_2_MSK
Reset	0	0	0	0	0 0	0	0

Bits	8	9	11	12	13	14	15
R W	EB1_MSK	—		ER1_MSK	ER2_MSK	—	
Reset	0	0	0 0	0	0	0	0

Bits	16	18	19	21	22	23
R W	CEN		—		SBER_1	SBER_2
Reset	1	0	1 0	0 0	0	0

Bits	24	25	26	27	28	29	31
R W	EB1_PR		—		ER1_PR	—	
Reset	0	0	0	0	0	0 0	0

Figure 18-3. Serial DMA Mode Register (SDMR)

Table 18-2 describes the SDMR fields.

Table 18-2. SDMR Field Descriptions

Bits	Name	Description
0	GLB_1_MSK	Mask global mode on bus 1. 0 Mask global mode on bus 1 1 Enable global mode on bus 1
1	—	Reserved, should be cleared.
2	ADR_SEL	Address match bus select mode The decision for which bus the dma command is directed to is decided according to the External address of the DMA command and SDAQR and SDAQMR. 0 External bus select for dma command is according to BMR register in the UEC 1 External bus select for dma command is according address match.
2–5	—	Reserved, should be cleared.
6	BER_1_MSK	Mask bus 1 error events. 0 Mask bus 1 error events 1 Enable bus 1 error events
7	BER_2_MSK	Mask bus 2 error events. 0 Mask bus 2 error events 1 Enable bus 2 error events
8	EB1_MSK	Mask emergency on external bus 1. 0 Mask emergency towards External bus 1 1 Enable emergency towards External bus 1
9–11	—	Reserved, should be cleared.
12	ER1_MSK	Mask emergency on Multi-user RAM port 1. 0 Mask Multi-user RAM port 1 emergency 1 Enable Multi-user RAM port 1 emergency
13	ER2_MSK	Mask emergency on Multi-user RAM port 2. 0 Mask Multi-user RAM port 2 emergency 1 Enable Multi-user RAM port 2 emergency
14–15	—	Reserved, should be cleared.
16–18	CEN	SDMA Temporary Buffer Size: 000 512 Bytes 001 1 Kbytes 010 1.5 Kbytes 011 2 KBytes 100 2.5 Kbytes 101 3 KBytes 110 Reserved 111 Reserved
19–21	—	Reserved, should be cleared.
22	SBER_1	Stop at bus error event on bus 1. 0 Continue DMA transactions regularly, even if a bus error occurred on bus 1 1 Stop the DMA transactions (after ending all current open dma transactions) if a bus error occurred on bus 1

Table 18-2. SDMR Field Descriptions (continued)

Bits	Name	Description
23	SBER_2	Stop at bus error event on bus 2. 0 Continue DMA transactions regularly, even if a bus error occurred on bus 2 1 Stop the DMA transactions (after ending all current open dma transactions) if a bus error occurred on bus 2
24–25	EB1_PR	Set priority on bus 1 (Used when the SDMA is not in emergency state or when emergency requests are masked by EB1_MSK. When the SDMA is in emergency state, highest priority is used). 00 Level 0 (Lowest) 01 Level 1 10 Level 2 11 Level 3 (Highest)
26–27	—	Reserved, should be cleared.
28	ER1_PR	Set priority on Multi-user RAM port 1 (Used when the SDMA is not in emergency state or when emergency requests are masked by ER1_MSK. When the SDMA is in emergency state, highest priority is used). 0 Low priority. 1 High priority.
29	ER2_PR	Set priority on Multi-user RAM port 2 (Used when the SDMA is not in emergency state or when emergency requests are masked by ER2_MSK. When the SDMA is in emergency state, highest priority is used.). 0 Low priority. 1 High priority.
30–31	—	Reserved, should be cleared.

18.1.8.3 Serial DMA Threshold Registers (SDTR1 and SDTR2)

SDTR1 and SDTR2, shown in Figure 18-7, with a combination of SDHY1 and SDHY2 defines the high priority levels towards Multi-user RAM and External buses. High priority is activated when reaching the threshold value. The high priority will be held until reaching the hysteresis value.

When the internal DMA read data buffer (buffer for DMA read commands) reaches its threshold value (RBTH), a high priority indicator towards Multi-user RAM port will be asserted, and will be held asserted until the data buffer reaches its hysteresis value (RBHY).

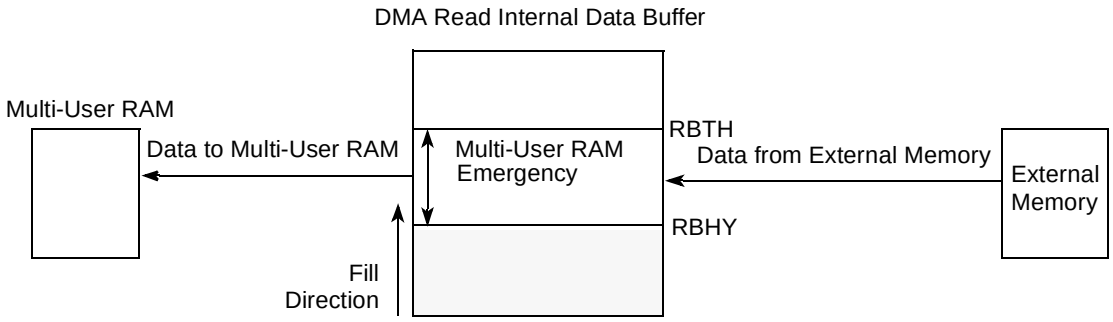


Figure 18-4. DMA Read Data Path

When the internal DMA write data buffer (buffer for DMA write commands) reaches its threshold value (WBTH), a high priority indicator towards the External bus will be asserted, and will be held asserted until the data buffer reaches the hysteresis value (WBHY).

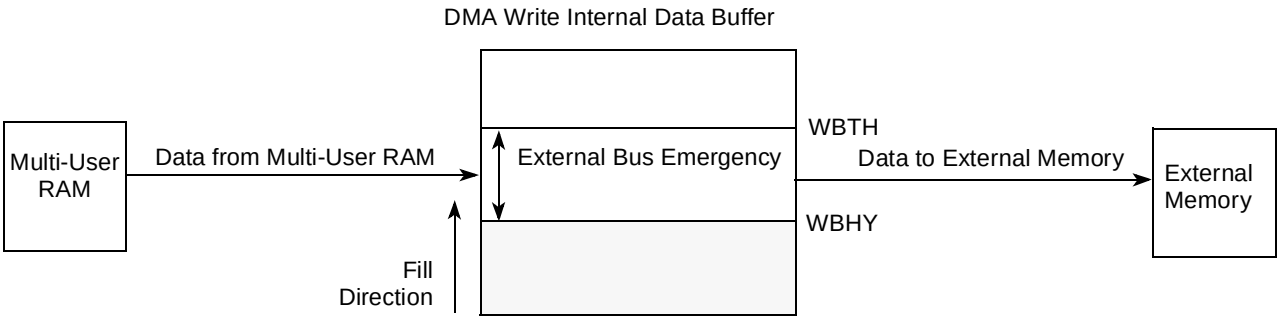


Figure 18-5. DMA Write Data Path

The DMA contains a command queue for each External bus. When the DMA command queue reaches it threshold (CQTH), a high priority indicator towards the Multi-user RAM port and External bus will be asserted, and will be held asserted until the command queue reaches the hysteresis value (CQHY).

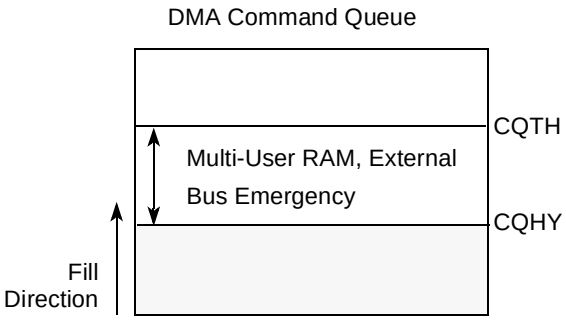


Figure 18-6. DMA Command Queue

Offset SDTR1: 0x4008
SDTR2: 0x400C

Access: Read/Write

Bits	0–2	3–7	8–9	10–15	16–25	26–31
R/W	—	CQTH	—	RBTH	—	WBTH
Reset	0 0 0	0 1 0 0 1	0 0	1 1 0 0 0 0	0 0 0 0 0 0 0 0 0 0	0 0 1 1 0 0 0 0

Figure 18-7. Serial DMA Threshold Registers (SDTR1 and SDTR2)

Table 18-3 describes the SDTRx fields.

Table 18-3. SDTRx Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3–7	CQTH	Command queue threshold. When the command queue reaches its threshold value, a high priority indication towards Multi-user RAM port and external address bus will be asserted

Table 18-3. SDTRx Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved, should be cleared.
10–15	RBTH	Read internal data buffer threshold. When the Read internal data buffer reaches its threshold value, a high priority indication towards Multi-user RAM port will be asserted
16–25	—	Reserved, should be cleared.
26–31	WBTH	Write internal data buffer threshold. When the Write internal data buffer reaches its threshold value, a high priority indication towards the External bus will be asserted

18.1.8.4 Serial DMA Hysteresis Registers (SDHY1 and SDHY2)

SDHY1 and SDHY2, shown in Figure 18-8, with a combination of SDTR1 and SDTR2 defines the high priority levels towards Multi-user RAM and External buses. High priority is activated when the parameter (internal data buffer or command queue in this case) reaches the threshold values, found in the SDTR*x* registers. The high priority indicator will be held until the parameter reaches the hysteresis value, found in the SDHY*x* registers. For a full explanation, see Section 18.1.8.3, "Serial DMA Threshold Registers (SDTR1 and SDTR2)."

Offset SDHY1: 0x4010
SDHY2: 0x4014

Access: Read/Write

Bits	0–2	3–7	8–9	10–15	16–25	26–31
R W	—	CQHY	—	RBHY	—	WBHY
Reset	0 0 0	0 1 1 0	0 0	1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0

Figure 18-8. Serial DMA Hysteresis Registers (SDHY1 and SDHY2)

Table 18-4 describes the SDHYx fields.

Table 18-4. SDHYx Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3–7	CQHY	Command queue hysteresis. When the command queue reaches its threshold value (CQTH), a high priority indicator towards Multi-user RAM port and external address bus will be asserted. It will be held until reaching the hysteresis value stored in these bits (CQHY).
8–9	—	Reserved, should be cleared.
10–15	RBHY	Read internal data buffer hysteresis. When the Read internal data buffer reaches its threshold value (RBTH), a high priority indicator towards Multi-user RAM port will be asserted. It will be held until reaching the hysteresis value stored in these bits (RBHY).

Table 18-4. SDHYx Field Descriptions (continued)

Bits	Name	Description
16–25	—	Reserved, should be cleared.
26–31	WBHY	Write internal data buffer hysteresis. When the Write internal data buffer reaches its threshold value (WBTH), a high priority indicator towards the External bus will be asserted. It will be held until reaching the hysteresis value stored in these bits (WBHY).

18.1.8.5 Serial DMA Transfer Address Registers (SDTA1 and SDTA2)

SDTA1 and SDTA2, shown in Figure 18-9, are read only registers, that hold the address accessed during the current bus transaction (bus number 1 and bus number 2 respectively). In case of bus error, the address is locked in the register until the corresponding status bit in the SDSR register is reset.

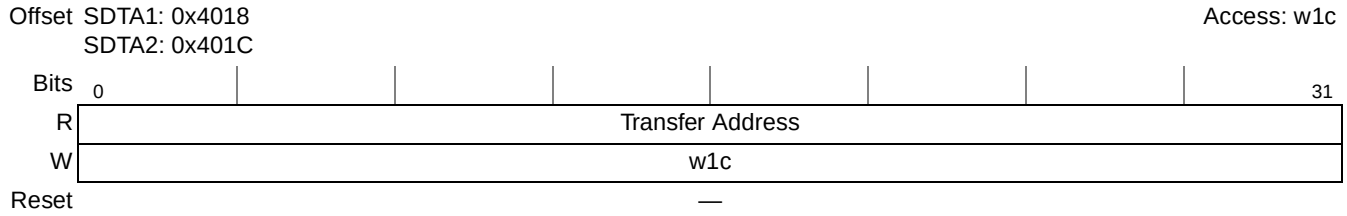


Figure 18-9. Serial DMA Transfer Address Registers (SDTA1 and SDTA2)

Table 18-5 describes the SDTAx fields.

Table 18-5. SDTAx Field Descriptions

Bits	Name	Description
0–31	Transfer Address	Address accessed during current bus transaction. The value is updated at every transfer end. The value is locked in case of bus error and released when the SDSR[BER_x] bit is reset by the user (writing ‘1’ to it).

18.1.8.6 Serial DMA Transfer Communication Channel Number Registers (SDTM1 and SDTM2)

SDTM1 and SDTM2, shown in Figure 18-10, are read only registers that hold the serial number of the peripheral that was served during the current bus transaction (bus number 1 and bus number 2 respectively). In case of bus error, the MSNUM value is locked in the register until the corresponding status bit in the SDSR register is reset.

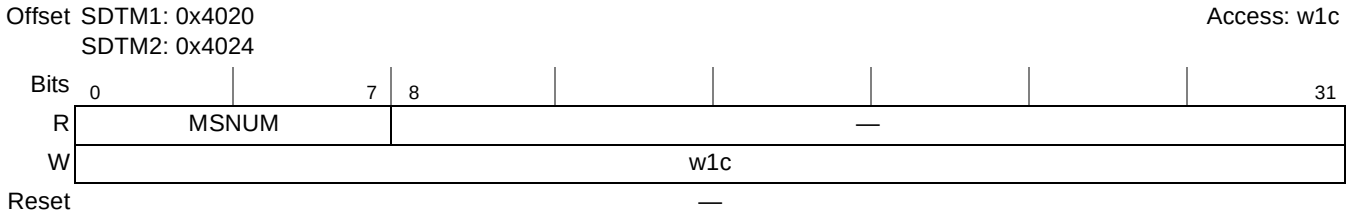


Figure 18-10. Serial DMA Transfer MSNUM Registers (SDTM1 and SDTM2)

Table 18-6 describes the SDTM*x* fields.

Table 18-6. SDTM*x* Field Descriptions

Bits	Name	Description
0–7	MSNUM	MSNUM served during current bus transaction. The value is updated at every transfer end. The value is locked in case of bus error and released when the SDSR[BER_x] bit is reset by the user (writing ‘1’ to it).
8–31	—	Reserved, should be cleared.

18.1.8.7 Serial DMA Address Qualify Registers (SDAQR)

SDAQR, shown in Figure 18-11, holds the 16 msb address value according to which the address match will be carried out. Address match (for which the External bus does the current DMA transaction) is done according the external address of the DMA command and the values of SDAQR and SDAQMR. The match is done only on the 16 msb’s of the external address. SDQAR[BS] defines the bus (bus1 or bus2) in case of address match.

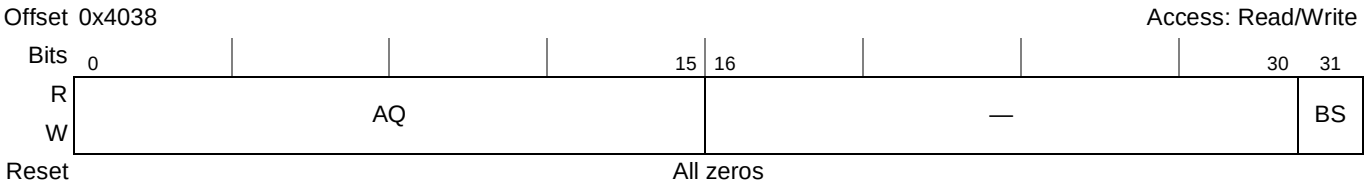


Figure 18-11. Serial DMA Address Qualify Register (SDAQR)

Table 18-7 describes the SDAQR fields.

Table 18-7. SDAQR Field Descriptions

Bits	Name	Description
0–15	AQ	Address Qualifier. 16 msbs of the address match.
16–30	—	Reserved, should be cleared.
31	BS	Bus select in case of address match 0 Bus2 (if match), bus1 (if no match) 1 Bus1(if match), bus2 (if no match)

18.1.8.8 Serial DMA Address Qualify Mask Register (SDAQMR)

SDAQMR, shown in Figure 18-12, hold the 16 msb's of the address mask value according to which the address match will be done. Address match (for which the External bus does the current DMA transaction) is done according the external address of the DMA command and the values of SDAQR and SDAQMR. The match is done only on the 16 msb's of the external address. SDQAR[BS] defines the bus (bus1 or bus2) in case of address match.

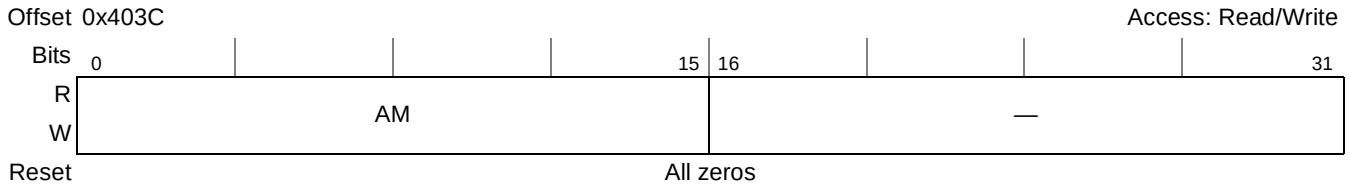


Figure 18-12. Serial DMA Address Qualify Mask Register (SDAQMR)

Table 18-8 describes the SDAQMR fields.

Table 18-8. SDAQMR Field Descriptions

Bits	Name	Description
0–15	AM	16 msb bits of the address mask. 1 Mask on this bit (no match check on this bit). 0 Match check on this bit.
16–31	—	Reserved, should be cleared.

18.1.8.9 Serial DMA Temporary Buffer Base in Multi-User RAM Value (SDEBCR)

SDEBCR, shown in Figure 18-13, holds the Temporary Buffer base address in multi-user RAM.

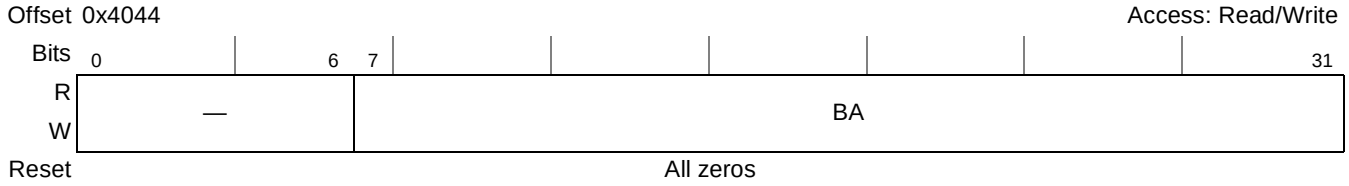


Figure 18-13. Serial DMA Temporary Buffer Base in Multi-User RAM Value (SDEBCR)

Table 18-9 describes the SDEBCR fields.

Table 18-9. SDEBCR Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7–31	BA	Temporary Buffer base address in Multi-user RAM. The address must be 4KB aligned.

18.2 Interrupt Controller

The peripherals of the QUICC Engine module (UCCs, USB, SDMA, Timers, and SPIs where applicable) are the main interrupt sources. It is possible to program a priority hierarchy, and assign a unique identifier (vector) to the interrupts from these sources. The QUICC Engine module offers two priority schemes—a grouped hierarchy, and a spread hierarchy.

18.2.1 Interrupt Configuration

Figure 18-14 shows the QUICC Engine interrupt structure. The interrupt controller receives interrupts from various internal sources within the QUICC Engine module.

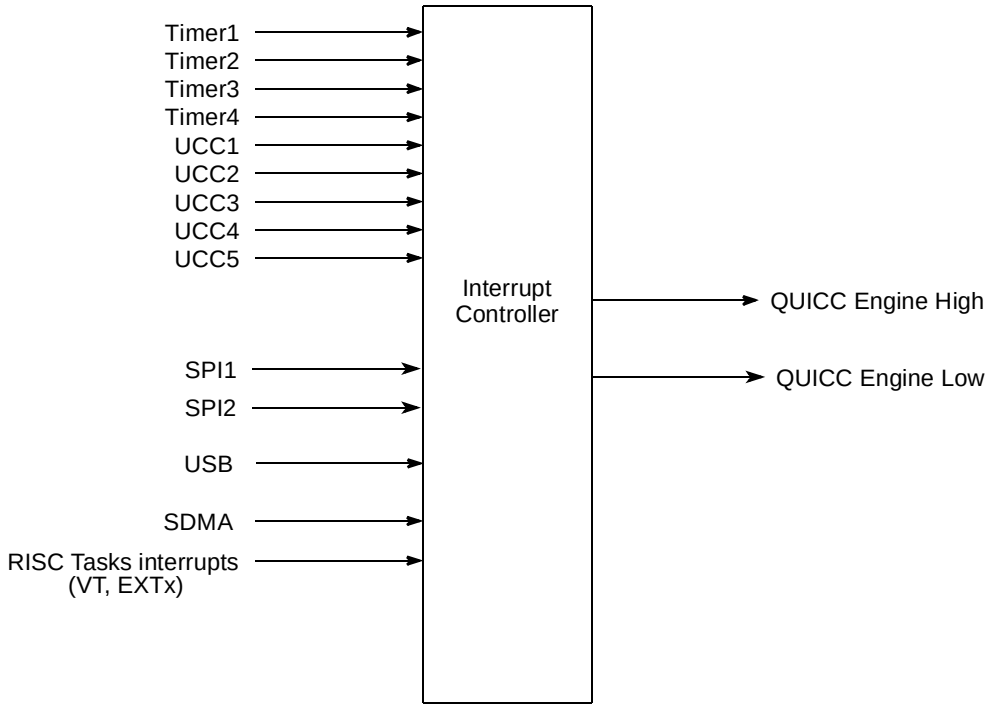


Figure 18-14. QUICC Engine Module Interrupt Structure

The interrupt controller allows masking of each interrupt source. In addition, multiple events within a QUICC Engine module peripheral event are also maskable. The Interrupt controller can be programmed to split the interrupts between two interrupt outputs: QUICC Engine High and QUICC Engine Low.

All interrupt sources are prioritized and bits are set in the interrupt pending register (CIPNR). On the QUICC Engine module, the prioritization of the interrupt sources can be programmed in two areas:

The relative priority of the UCCs, USB, and SPIs. See Section 18.2.3, “UCC Relative Priority,” for more information.

- Assigning highest priority to one interrupt source. See Section 18.2.4, “Highest Priority Interrupt,” for more information.

Each interrupt output (QUICC Engine High, QUICC Engine Low) has its corresponding interrupt vector register. The QUICC Engine interrupt vector register (CIVEC) and QUICC Engine High interrupt vector register (CHIVEC) are updated with a 6-bit vector corresponding to the peripheral with the current highest priority within the group assigned to each interrupt output.

18.2.2 Interrupt Source Priorities

The interrupt controller has 14 interrupt sources that assert two interrupt requests outputs (QUICC Engine High and QUICC Engine Low). The user is able to allocate some interrupt sources to the QUICC Engine High interrupt output (see Section 18.3.2, "QUICC Engine System Interrupt Control Register (CICNR)"). All the sources can be allocated to the QUICC Engine Low interrupt output. Table 18-10 shows prioritization of all interrupt sources. As described in the following sections, flexibility exists in the relative ordering of the interrupts, but, in general, relative priorities are not shown. Each interrupt priority number corresponds to each table entry.

Note that the group and spread options, shown with XCC and YCC entries in Table 18-10, are described in Section 18.2.3, "UCC Relative Priority."

Table 18-10. Interrupt Source Priority Levels

Priority Level	Interrupt Source Description	Multiple Events
1	Highest	—
2	MIXA1 (Grouped/Spread)	Yes
3	MIXA2 (Grouped)	Yes
4	MIXA3 (Grouped)	Yes
5	MIXA4 (Grouped)	Yes
6	RTB1 (Spread)	Yes
7	XCC1 (Grouped)	Yes
8	XCC2 (Grouped)	Yes
9	XCC3 (Grouped)	Yes
10	XCC4 (Grouped)	Yes
11	MIXA2 (Spread)	Yes
12	XCC5 (Grouped)	Yes
13	XCC6 (Grouped)	Yes
14	XCC7 (Grouped)	Yes
15	XCC8 (Grouped)	Yes
16	RTB1 (Grouped)	Yes
17	RTB2 (Grouped)	Yes
18	RTB3 (Grouped)	Yes
19	RTB4 (Grouped)	Yes

Table 18-10. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
20	RTB2 (Spread)	Yes
21	YCC1 (Grouped)	Yes
22	YCC2 (Grouped)	Yes
23	YCC3 (Grouped)	Yes
24	YCC4 (Grouped)	Yes
25	MIXA3 (Spread)	Yes
26	YCC5 (Grouped)	Yes
27	YCC6 (Grouped)	Yes
28	YCC7 (Grouped)	Yes
29	YCC8 (Grouped)	Yes
30	MIXA5 (Grouped)	Yes
31	MIXA6 (Grouped)	Yes
32	MIXA7 (Grouped)	Yes
33	MIXA8 (Grouped)	Yes
34	RTB3 (Spread)	Yes
35	WCC1 (Grouped)	Yes
36	WCC2 (Grouped)	Yes
37	WCC3 (Grouped)	Yes
38	WCC4 (Grouped)	Yes
39	MIXA4 (Spread)	Yes
40	WCC5 (Grouped)	Yes
41	WCC6 (Grouped)	Yes
42	WCC7 (Grouped)	Yes
43	WCC8 (Grouped)	Yes
44	RTB5 (Grouped)	Yes
45	RTB6 (Grouped)	Yes
46	RTB7 (Grouped)	Yes
47	RTB8 (Grouped)	Yes
48	RTB4 (Spread)	Yes
49	ZCC1 (Grouped)	Yes
50	ZCC2 (Grouped)	Yes
51	ZCC3 (Grouped)	Yes
52	ZCC4 (Grouped)	Yes

Table 18-10. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
53	MIXA5 (Spread)	Yes
54	ZCC5 (Grouped)	Yes
55	ZCC6 (Grouped)	Yes
56	ZCC7 (Grouped)	Yes
57	ZCC8 (Grouped)	Yes
58	MIXA5 (Spread)	Yes
59	Reserved	—
60	XCC1 (Spread)	Yes
61	YCC1 (Spread)	Yes
62	Reserved	—
63	WCC1 (Spread)	Yes
64	ZCC1 (Spread)	Yes
65–66	Reserved	—
67	MIXA6 (Spread)	Yes
68	Reserved	—
69	XCC2 (Spread)	Yes
70	YCC2 (Spread)	Yes
71	Reserved	—
72	WCC2 (Spread)	Yes
73	ZCC2 (Spread)	Yes
74–75	Reserved	—
76	RTB6 (Spread)	Yes
77	Reserved	—
78	XCC3 (Spread)	Yes
79	YCC3 (Spread)	Yes
80	Reserved	—
81	WCC3 (Spread)	Yes
82	ZCC3 (Spread)	Yes
83–84	Reserved	—
85	MIXA7 (Spread)	Yes
86	Reserved	—
87	XCC4 (Spread)	Yes
88	YCC4 (Spread)	Yes

Table 18-10. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
89	Reserved	—
90	WCC4 (Spread)	Yes
91	ZCC4 (Spread)	Yes
92–93	Reserved	—
94	RTB7 (Spread)	Yes
95	Reserved	—
96	XCC5 (Spread)	Yes
97	YCC5 (Spread)	Yes
98	Reserved	—
99	WCC5 (Spread)	Yes
100	ZCC5 (Spread)	Yes
101–102	Reserved	—
103	MIXA8 (Spread)	Yes
104	Reserved	—
105	XCC6 (Spread)	Yes
106	YCC6 (Spread)	Yes
107	Reserved	—
108	WCC6 (Spread)	Yes
109	ZCC6 (Spread)	Yes
110–111	Reserved	—
112	RTB8 (Spread)	Yes
113	Reserved	—
114	XCC7 (Spread)	Yes
115	YCC7 (Spread)	Yes
116	Reserved	—
117	WCC7 (Spread)	Yes
118	ZCC7 (Spread)	Yes
119–121	Reserved	—
122	XCC8 (Spread)	Yes
123	YCC8 (Spread)	Yes
124	Reserved	—
125	WCC8 (Spread)	Yes
126	ZCC8 (spread)	Yes

There are two ways to program the flexibility of the QUICC Engine module interrupt priorities:

- The UCC relative priority option. See Section 18.2.3, “UCC Relative Priority,” for more information.
- The highest priority option. See Section 18.2.4, “Highest Priority Interrupt,” for more information.

18.2.3 UCC Relative Priority

The relative priority between the five UCCs is programmable and can be changed dynamically. There is no entry for UCC1-4 and UCC5 in Table 18-10, but rather there are entries for XCC1–XCC8 and YCC1–YCC8. UCC5 and UCC1-4 can be mapped to any YCC location and any XCC location, respectively. The UCC priorities are programmed in the QUICC Engine interrupt priority registers (CIPXCC and CIPYCC) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the XCC and YCC entries has the following two options:

- Group—In the group scheme, all UCCs are grouped together at the top of the priority table, ahead of most other QUICC Engine interrupt sources. This scheme is ideal for applications where all UCCs 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 that the other sources can have lower interrupt latencies. This scheme is also programmed in the CICR but cannot be changed dynamically.

NOTE: On Backward Compatibility

The allocation of the interrupts allows for backward compatibility with the 82xx/85xx family of devices that are listed in Table 18-11.

Table 18-11. Mapping of FCCs and SCCs to UCCs for 82xx/85xx Compatibility

82xx Device	QUICC Engine Device
FCC1	UCC1
FCC2	UCC2
FCC3	UCC3
SCC1	UCC5

18.2.4 Highest Priority Interrupt

In addition to the UCC relative priority option, CICR[HP] can be used to specify one interrupt source based on its highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but it is serviced before any other interrupt in the table.

CICR[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.

18.2.5 Masking Interrupt Sources

By programming the system interrupt mask register (CIMR), the user can mask interrupt requests to the core. Each CIMR bit corresponds to an interrupt source. The corresponding CIMR bit is set to enable an interrupt. When a masked interrupt source has a pending interrupt request, the corresponding CIPNR bit is set, even though the interrupt is not sent to the system interrupt controller. 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 18-10 shows the interrupt sources that have multiple interrupting events. Figure 18-15 shows an example on how the masking occurs, using a UCC.

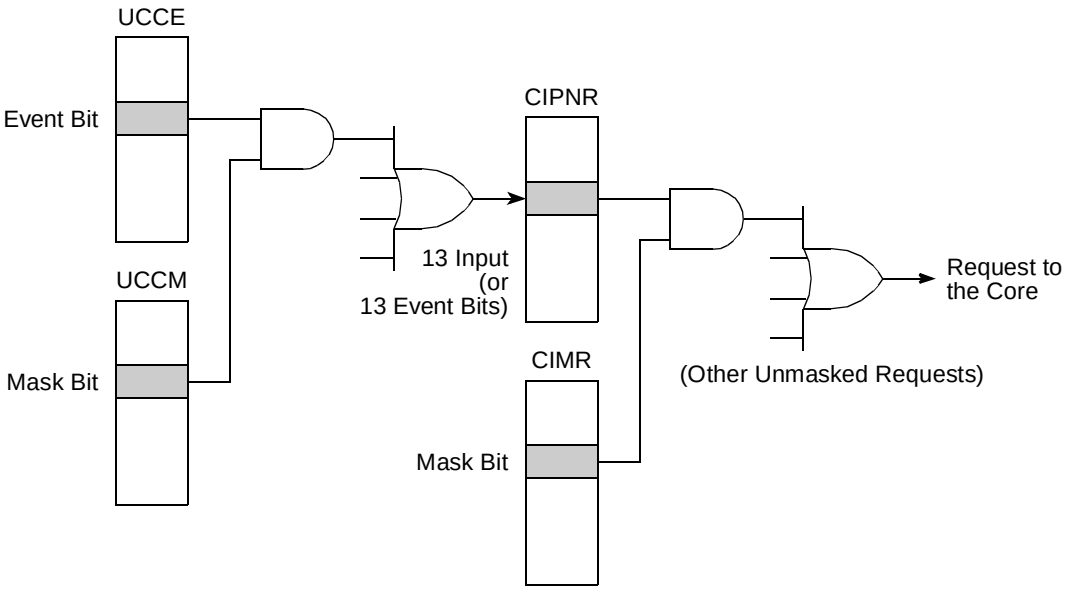


Figure 18-15. Interrupt Request Masking

18.2.6 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core processor according to priority. The interrupt vector that allows the core processor to locate the interrupt service routine is made available to the core processor by reading CIVEC for QUICC Engine Low interrupt output and CHIVEC for QUICC Engine High interrupt output. The interrupt controller passes an interrupt vector according to the highest-priority, unmasked, pending interrupt. Table 18-12 lists encodings for the six low-order bits of the interrupt vector.

Table 18-12. Encoding the Interrupt Vector

Interrupt Number[1]	Interrupt Source Description	Interrupt Vector
0	Error (No interrupt)	0b00_0000
1	SPI2	0b00_0001
2	SPI1	0b00_0010
3	RTT	0b00_0011

Table 18-12. Encoding the Interrupt Vector (continued)

Interrupt Number[1]	Interrupt Source Description	Interrupt Vector
4	Reserved	0b00_0100
5	Reserved	0b00_0101
6	Reserved	0b00_0110
7	Reserved	0b00_0111
8	Reserved	0b00_1000
9	Reserved	0b00_1001
10	SDMA	0b00_1010
11	USB	0b00_1011
12	Timer1	0b00_1100
13	Timer2	0b00_1101
14	Timer3	0b00_1110
15	Timer4	0b00_1111
16–19	Reserved	0b01_0000–01_0011
20	VT	0b01_0100
21–24	Reserved	0b01_0101–0b01_1000
25	EXT1	0b01_1001
26	EXT2	0b01_1010
27	EXT3	0b01_1011
28	EXT4	0b01_1100
29–31	Reserved	0b01_1101–0b01_1111
32	UCC1	0b10_0000
33	UCC2	0b10_0001
34	UCC3	0b10_0010
35	UCC4	0b10_0011
36	Reserved	0b10_0100
37	Reserved	0b10_0101
38	Reserved	0b10_0110
39	Reserved	0b10_0111
40	UCC5	0b10_1000
41	Reserved	0b10_1001
42	Reserved	0b10_1010
43	Reserved	0b10_1011
44	Reserved	0b10_1100
45	Reserved	0b10_1101

Table 18-12. Encoding the Interrupt Vector (continued)

Interrupt Number[1]	Interrupt Source Description	Interrupt Vector
46	Reserved	0b10_1110
47	Reserved	0b10_1111
48	Reserved	0b11_0000
49	Reserved	0b11_0001
50–63	Reserved	0b11_0010–0b11_1111

1 The interrupt number is used in the HP field of the CICR register.

Note that the interrupt vector table differs from the interrupt priority table in the following two ways:

- The vectors are fixed; they are not affected by the group mode, spread mode, or the relative priority order of the peripherals.
- An error vector exists as the first entry in Table 18-12. The error vector is issued when no interrupt is requesting service.

18.3 Programming Model

The interrupt controller registers are described in the following sections:

18.3.1 QUICC Engine System Interrupt Configuration Register (CICR)

CICR, shown in Figure 18-16, defines the highest priority interrupt and whether interrupts are grouped or spread according to the priority table, Table 18-10.

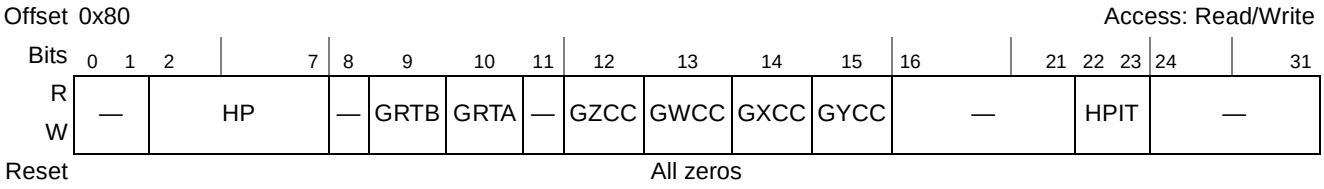


Figure 18-16. QUICC Engine System Interrupt Configuration Register (CICR)

The CICR register bits are described in Table 18-13.

Table 18-13. CICR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, must be cleared.
2–7	HP	Highest priority. These bits specify the 6-bit interrupt number of the single interrupt controller interrupt source that is advanced in the table based on the highest priority. HP can be modified dynamically. To retain the original priority, HP is programed to the interrupt number assigned to MIXA1.
8	—	Reserved, should be cleared.
9	GRTB	RTB (RISC Tasks B interrupts) Priority Scheme. GRTB selects the relative RTB priority scheme and cannot be changed dynamically. 0 Grouped. The RTBs are grouped by priority at the top of the table. 1 Spread. The RTBs are spread in the table according to priority.
10	GRTA	RTA (RISC Tasks A interrupts) Priority Scheme. GRTA selects the relative RTA priority scheme and cannot be changed dynamically. 0 Grouped. The RTAs are grouped by priority at the top of the table. 1 Spread. The RTAs are spread in the table according to priority.
11	—	Reserved, should be cleared.
12	GZCC	ZCC Priority Scheme. GZCC selects the relative ZCC priority scheme and cannot be changed dynamically. 0 Grouped. The ZCCs are grouped by priority at the top of the table. 1 Spread. The ZCCs are spread in the table according to priority.
13	GWCC	WCC Priority Scheme. GWCC selects the relative WCC priority scheme and cannot be changed dynamically. 0 Grouped. The WCCs are grouped by priority at the top of the table. 1 Spread. The WCCs are spread in the table according to priority.
14	GXCC	XCC Priority Scheme. GXCC selects the relative Group1 priority scheme and cannot be changed dynamically. 0 Grouped. The XCCs are grouped by priority at the top of the table. 1 Spread. The XCCs are spread in the table according to priority.
15	GYCC	YCC Priority scheme. GYCC selects the relative YCC priority scheme and cannot be changed dynamically. 0 Grouped. The YCCs are grouped by priority at the top of the table. 1 Spread. The YCCs are spread in the table according to priority.
16–21	—	Reserved, should be cleared.

Table 18-13. CICR Field Descriptions (continued)

Bits	Name	Description
22–23	HPIT	Highest Priority Interrupt position. Defines the interrupt output that is asserted by the highest priority interrupt. 00 QUICC Engine Low is asserted when the highest priority interrupt occurs. 01 Reserved. 10 QUICC Engine High is asserted when the highest priority interrupt occurs. 11 Reserved.
24–31	—	Reserved, should be cleared.

18.3.2 QUICC Engine System Interrupt Control Register (CICNR)

CICNR, shown in Figure 18-17, defines the output interrupt type (QUICC Engine High or QUICC Engine Low) in the XCC1, XCC2, YCC1, YCC2, WCC1, WCC2, ZCC1, and ZCC2 priority positions. All other priority positions will assert QUICC Engine Low signal unless they are selected as highest priority interrupt (see bit 22 in Section 18.3.1, "QUICC Engine System Interrupt Configuration Register (CICR)").

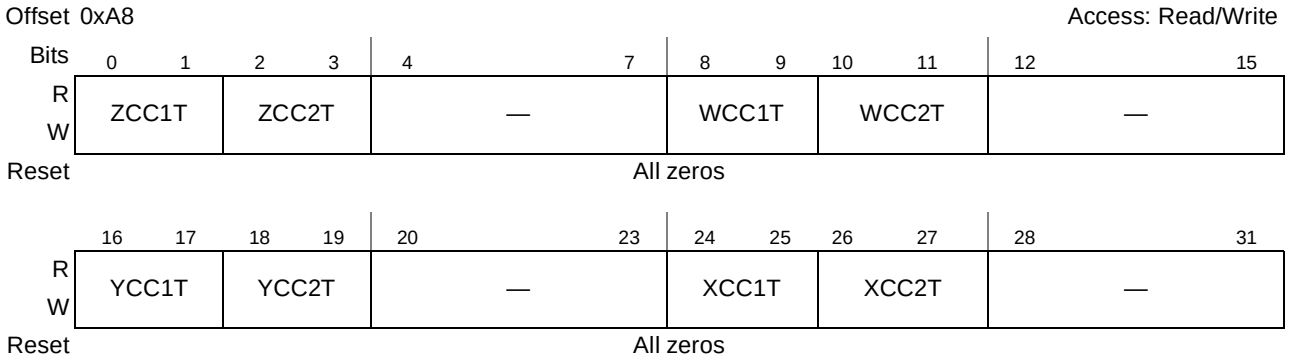


Figure 18-17. QUICC Engine System Internal Interrupt Control Register (CICNR)

Table 18-14 defines the bit fields of CICNR.

Table 18-14. CICNR Bit Settings

Bits	Name	Description
0–1	ZCC1T	ZCC1 priority position output interrupt Type. Defines which type of the output interrupt signal (QUICC Engine High or QUICC Engine Low) asserts its request to the system interrupt controller in the ZCC1 priority position. These bits can not be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it will not happen during the change). The definition of ZCC1T is as follows: 00 QUICC Engine Low request is asserted to the system interrupt controller for ZCC1. 01 Reserved. 10 QUICC Engine High request is asserted to the system interrupt controller for ZCC1. 11 Reserved.
2–3	ZCC2T	Same as ZCC1T, but for ZCC2T.
4–7	—	Write ignored, read = 0

Table 18-14. CICNR Bit Settings (continued)

Bits	Name	Description
8–9	WCC1T	WCC1 priority position output interrupt Type. Defines which type of the output interrupt signal (QUICC Engine High or QUICC Engine Low) asserts its request to the system interrupt controller in the WCC1 priority position. These bits can not be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it will not happen during the change). The definition of WCC1T is as follows: 00 QUICC Engine Low request is asserted to the system interrupt controller for WCC1. 01 Reserved. 10 QUICC Engine High request is asserted to the system interrupt controller for WCC1. 11 Reserved
10–11	WCC2T	Same as WCC1T, but for WCC2T.
12–15	—	Write ignored, read = 0
16–17	YCC1T	YCC1 priority position output interrupt Type. Defines which type of the output interrupt signal (QUICC Engine High or QUICC Engine Low) asserts its request to the system interrupt controller in the YCC1 priority position. These bits can not be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it will not happen during the change). The definition of YCC1T is as follows: 00 QUICC Engine Low request is asserted to the system interrupt controller for YCC1. 01 Reserved. 10 QUICC Engine High request is asserted to the system interrupt controller for YCC1. 11 Reserved
18–19	YCC2T	Same as YCC1T, but for YCC2T.
20–23	—	Write ignored, read = 0
24–25	XCC1T	XCC1 priority position output interrupt Type. Defines which type of the output interrupt signal (QUICC Engine High or QUICC Engine Low) asserts its request to the system interrupt controller in the XCC1 priority position. These bits can not be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it will not happen during the change). The definition of XCC1T is as follows: 00 QUICC Engine Low request is asserted to the system interrupt controller for XCC1. 01 Reserved. 10 QUICC Engine High request is asserted to the system interrupt controller for XCC1. 11 Reserved
26–27	XCC2T	Same as XCC1T, but for XCC2T.
28–31	—	Write ignored, read = 0

18.3.3 QUICC Engine System RISC Interrupts Control Register (CRICR)

CRICR, shown in Figure 18-18, defines the interrupt output type (QUICC Engine High or QUICC Engine Low) for RISC tasks interrupts.

Offset 0xBC Access: Read/Write

Bits	0–1	2–3	4–7	8–9	10–11	12–31
R	RTB1T	RTB2T	—	RTA1T	RTA2T	—
W						
Reset	All zeros					

Figure 18-18. QUICC Engine System RISC Interrupts Control Register (CRICR)

Table 18-15 defines the bit fields of CRICR.

Table 18-15. CRICR Bit Settings

Bits	Name	Description
0–1	RTB1T	RTB1 priority position output interrupt Type. Defines which type of the output interrupt signal (QUICC Engine High or QUICC Engine Low) asserts its request to the system interrupt controller in the RTB1 priority position. These bits can not be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it will not happen during the change). The definition of RTB1T is as follows: 00 QUICC Engine Low request is asserted to the system interrupt controller for RTB1. 01 Reserved. 10 QUICC Engine High request is asserted to the system interrupt controller for RTB1. 11 Reserved
2–3	RTB2T	Same as RTB1T, but for RTB2T.
4–7	Reserved	Reserved, should be cleared.
8–9	RTA1T	RTA1 priority position output interrupt Type.Defines which type of the output interrupt signal (QUICC Engine High or QUICC Engine Low) asserts its request to the system interrupt controller in the RTA1 priority position. These bits can not be changed dynamically. (If software really wants to change it, it has to make sure the corresponding interrupt source is masked or it will not happen during the change). The definition of RTA1T is as follows: 00 QUICC Engine Low request is asserted to the system interrupt controller for RTA1. 01 Reserved. 10 QUICC Engine High request is asserted to the system interrupt controller for RTA1. 11 Reserved
10–11	RTA2T	Same as RTA1T, but for RTA2T.
12–31	Reserved	Reserved, should be cleared.

18.3.4 QUICC Engine System Interrupt Priority Register for WCC Peripherals (CIPWCC)

CIPWCC, shown in Figure 18-19, defines the priority between Timers, SPIs, and external RISC interrupts.

Offset 0x98 Access: Read/Write

Bits	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R W	WCC1	WCC2	WCC3	WCC4	—	WCC5	WCC6	WCC7	WCC8	—
Reset	0 0 0	0	0 1 0	1	0 0 1 1	0 0 0 0	1 0 0 1	0 1 1 1	0 1 1 1	0 0 0 0

Figure 18-19. QUICC Engine System Interrupt Priority Register for WCC (CIPWCC)

The CIPWCC register bits are described in Table 18-16.

Table 18-16. CIPWCC Field Descriptions

Bits	Name	Description
0–2	WCC1	WCC Priority order. These bits define which QUICC Engine Timer SPI and SDMA error asserts its request in the WCC1 priority position. Note that the same peripheral must not be programmed to more than one priority position (1–8). These bits can be changed dynamically. 000 SPI2 asserts its request in the WCC1 position. 001 SPI1 asserts its request in the WCC1 position. 010 RTT asserts its request in the WCC1 position. 011 Reserved position is not active. 100 Reserved position is not active. 101 Reserved position is not active. 110 Reserved position is not active. 111 Reserved position is not active
3–11, 16–27	WCC2–WCC8	Same as WCC1, but for WCC2–WCC8.
12–15, 28–31	—	Reserved, should be cleared.

NOTE

The lack of SDMA interrupt sources are reported through each individual UCC or SPI channel. The SDMA channel bus error entry is the only true SDMA interrupt source that is reported when a bus error occurs during an SDMA access.

18.3.5 QUICC Engine System Interrupt Priority Register for XCC Peripherals (CIPXCC)

The QUICC Engine system interrupt priority for XCC peripherals register (CIPXCC), shown in Figure 18-20, defines the priorities between the UCCs.

Offset 0x90 Access: Read/Write

Bits	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R/W	XCC1	XCC2	XCC3	XCC4	—	XCC5	XCC6	XCC7	XCC8	—
Reset	0 0 0	0 1 0	1 0 0	1 1	0 0 0 0	1 0 0	1 0 1	1 0 1	1 0 0	0 0 0 0

Figure 18-20. QUICC Engine System Interrupt Priority Register for XCC (CIPXCC)

Table 18-17 describes CIPXCC fields.

Table 18-17. CIPXCC Field Descriptions

Bits	Name	Description
0–2	XCC1	Priority order. These bits define which UCC asserts its request in the XCC1 priority position. Note that each UCC must not be programmed to more than one priority position (1–8). These bits can be changed dynamically. 000 UCC1 asserts its request in the XCC1 position. 001 UCC2 asserts its request in the XCC1 position. 010 UCC3 asserts its request in the XCC1 position. 011 UCC4 asserts its request in the XCC1 position. 100 Reserved position is not active. 101 Reserved position is not active. 110 Reserved position is not active. 111 Reserved position is not active.
3–11, 16–27	XCC2–XCC8	Same as XCC1, but for XCC2–XCC8
12–15, 28–31	—	Reserved, should be cleared.

18.3.6 QUICC Engine System Interrupt Priority Register for YCC Peripherals (CIPYCC)

CIPYCC, shown in Figure 18-21, defines the priority of the UCCs.

Offset 0x94 Access: Read/Write

Bits	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R/W	YCC1	YCC2	YCC3	YCC4	—	YCC5	YCC6	YCC7	YCC8	—
Reset	0 0 0	0 1 0	1 0 0	1 1	0 0 0 0	1 0 0	1 0 1	1 0 1	1 0 0	0 0 0 0

Figure 18-21. QUICC Engine System Interrupt Priority Register for YCC (CIPYCC)

Table 18-18 describes the CIPYCC fields.

Table 18-18. CIPYCC Field Descriptions

Bits	Name	Description
0–2	YCC1	Priority order. These bits define which UCC asserts its request in the YCC1 priority position. The same UCC must not be programmed to multiple priority positions. This field can be changed dynamically. 000 UCC5 asserts its request in the YCC1 position. 001 Reserved position is not active. 010 Reserved position is not active. 011 Reserved position is not active. 100 Reserved position is not active. 101 Reserved position is not active. 110 Reserved position is not active. 111 Reserved position is not active.
3–11, 16–27	YCC2–YCC8	Same as YCC1, but for YCC2–YCC8
12–15, 28–31	—	Reserved, should be cleared.

NOTE: On Backward-Compatibility

The CIPXCC register corresponds to the SCPRR_H register in the 85xx PQIII devices. The CIPYCC register corresponds to the SCPRR_L register in the 85xx PQIII devices.

18.3.7 QUICC Engine System Interrupt Priority Register for ZCC Peripherals (CIPZCC)

CIPZCC, shown in Figure 18-22, defines the priority among a few peripherals.

Offset 0x9C Access: Read/ Write

Bits	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R W	ZCC1	ZCC2	ZCC3	ZCC4	—	ZCC5	ZCC6	ZCC7	ZCC8	—
Reset	0 0 0 0	0 1 0 1	0 0 1 1	0 0 0 0	1 0 0 1	0 1 1 1	0 1 1 1	0 0 0 0		

Figure 18-22. QUICC Engine System Interrupt Priority Register for ZCC (CIPZCC)

The CIPZCC register bits are described in Table 18-19.

Table 18-19. CIPZCC Field Descriptions

Bits	Name	Description
0–2	ZCC1	Priority order. These bits define which QUICC Engine Timer and SDMA Sys Error asserts its request in the ZCC1 priority position. The same peripheral must not be programmed to more than one priority position (1–8). These bits can be changed dynamically. 000 Reserved 001 SDMA Sys asserts its request in the ZCC1 position. 010 USB Sys asserts its request in the ZCC1 position 011 Timer1 asserts its request in the ZCC1 position. 100 Timer2 asserts its request in the ZCC1 position. 101 Timer3 asserts its request in the ZCC1 position. 110 Timer4 asserts its request in the ZCC1 position. 111 Reserved position is not active.
3–11, 16–27	ZCC2–ZCC8	Same as ZCC1, but for ZCC2–ZCC8.
12–15, 28–311	—	Reserved, should be cleared.

18.3.8 QUICC Engine System Interrupt Priority Register for RISC Tasks A (CIPRTA)

CIPRTA, shown in Figure 18-23 defines the relative priority between the interrupt sources VT and SINT.

Offset 0xB0 Access: Read/ Write

Bits	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R W	RTA1	RTA2	RTA3	RTA4	—	RTA5	RTA6	RTA7	RTA8	—
Reset	0 0 0	0	1 0 1	0	0 1 1	0 0 0 0	1 0 0 1	0 1 1 1	0 1 1 1	0 0 0 0

Figure 18-23. QUICC Engine System Interrupt Priority Register for RISC Tasks A (CIPRTA)

The CIPRTA register bits are described in Table 18-20.

Table 18-20. CIPRTA Field Descriptions

Bits	Name	Description
0–2	RTA1	Priority order. These bits define which RISC task A asserts its request in the RTA1 priority position. The same task must not be programmed to more than one priority position (1–8). These bits can be changed dynamically. 000 Reserved position is not active. 001 Reserved position is not active. 010 Reserved position is not active. 011 Virtual task asserts its request in the RTA1 position. 100 Reserved position is not active. 101 Reserved position is not active. 110 Reserved position is not active. 111 Reserved position is not active.
3–11, 16–27	RTA2–RTA8	Same as RTA1, but for RTA2–RTA8.
12–15, 28–31	—	Reserved, should be cleared.

18.3.9 QUICC Engine System Interrupt Priority Register for RISC Tasks B (CIPRTB)

CIPRTB, shown in Figure 18-24 defines the relative priority among the interrupt sources EXT1, EXT2, EXT3 and EXT4.

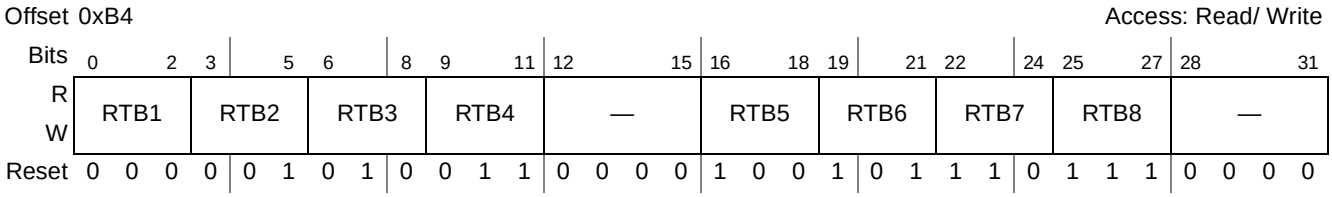


Figure 18-24. QUICC Engine System Interrupt Priority Register for RISC Tasks B (CIPRTB)

The CIPRTB register bits are described in Table 18-21.

Table 18-21. CIPRTB Field Descriptions

Bits	Name	Description
0–2	RTB1	Priority order. These bits define which RISC task B asserts its request in the RTB1 priority position. The same task must not be programmed to more than one priority position (1–8). These bits can be changed dynamically. 000 EXT1 asserts its request in the RTB1 position. 001 EXT2 asserts its request in the RTB1 position. 010 EXT3 asserts its request in the RTB1 position. 011 EXT4 asserts its request in the RTB1 position. 100 Reserved position is not active. 101 Reserved position is not active. 110 Reserved position is not active. 111 Reserved position is not active.
3–11, 16–27	RTB2–RTB8	Same as RTB1, but for RTB2–RTB8.
12–15, 28–31	—	Reserved, should be cleared.

18.3.10 QUICC Engine System Interrupt Pending Register (CIPNR)

Each bit in the CIPNR, shown in Figure 18-25, corresponds to an interrupt source. When an interrupt is received, the interrupt controller sets the corresponding CIPNR bit.

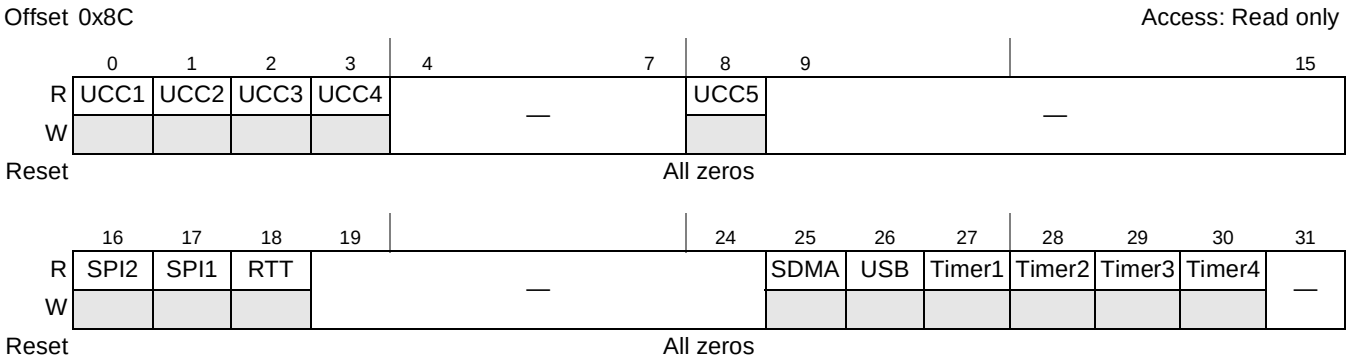


Figure 18-25. CIPNR Fields

CIPNR is a read-only register, so when a pending interrupt is handled, the user must clear the corresponding CIPNR bit by clearing the corresponding event register bit. The CIPNR bit position is fixed, and is not affected by the interrupt priority scheme.

18.3.11 QUICC Engine System Interrupt Mask Register (CIMR)

Each bit in the CIMR, shown in Figure 18-26, corresponds to an interrupt source. The user can mask an interrupt by clearing the relevant bit. Also, an interrupt can be enabled by setting the corresponding CIMR bit. When a masked interrupt occurs, the corresponding CIPNR bit is set, regardless of the CIMR bit although no interrupt request is passed to the core processor.

If an interrupt source requests an interrupt service when the user clears its CIMR bit, the request stops. If the user sets the CIMR bit later, a previously pending interrupt request is processed by the core processor, according to its assigned priority. The CIMR can be read by the user at any time.

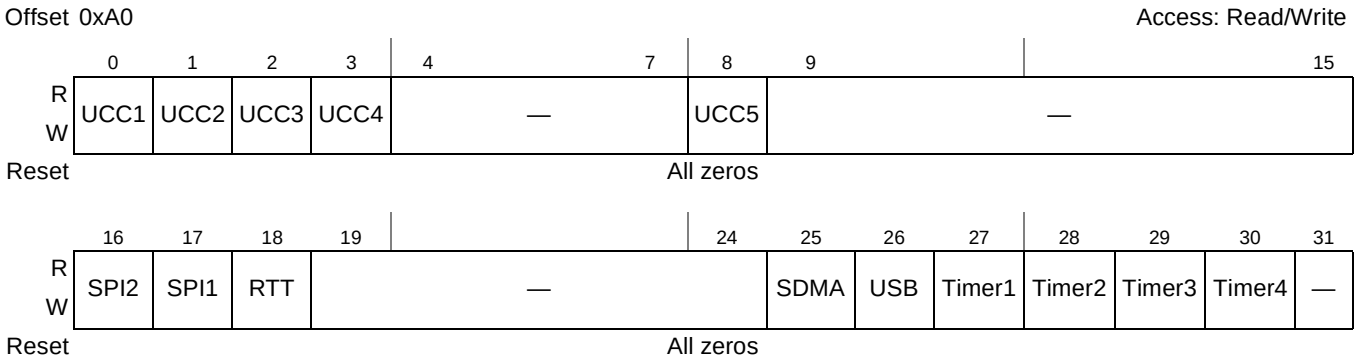


Figure 18-26. CIMR Field

NOTE

The CIMR bit positions are not affected by the relative interrupt priority. The user can clear pending register bits set by multiple interrupt events only by clearing *all* unmasked events in the corresponding event register. If a CIMR bit is masked at the same time that the corresponding CIPNR bit causes an interrupt request to the core processor, the error vector is issued (if no other interrupts are pending). Thus, the user must always include an error vector routine, even if it contains only an **rfi** instruction. The error vector cannot be masked.

18.3.12 QUICC Engine RISC Interrupt Pending Register (CRIPNR)

Each bit in the QUICC Engine RISC Interrupt Pending Register, represents a pending interrupt in the RISC. For each bit there is a corresponding event register described in Chapter 19, “Configuration.” Figure 18-27 shows the CRIPNR fields.

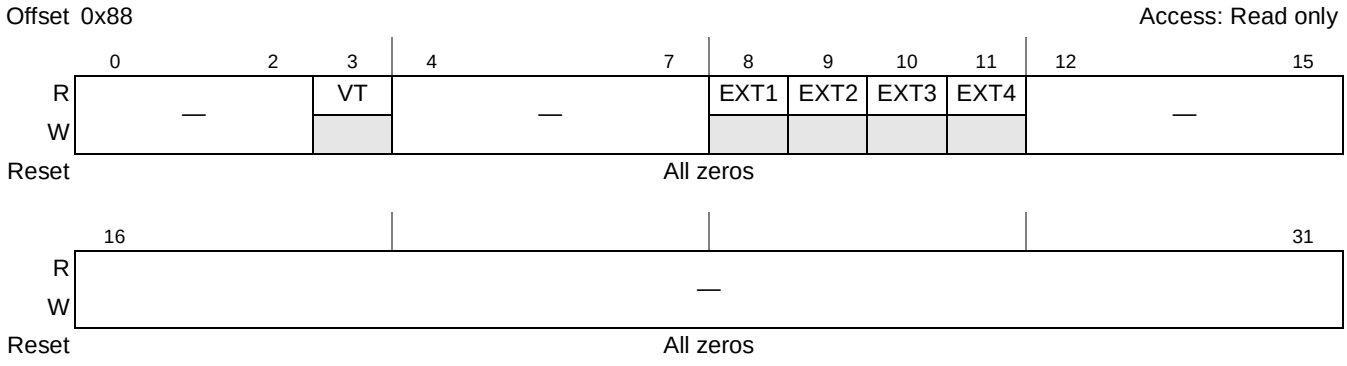


Figure 18-27. CRIPNR Fields

Following is a short description of the RISC interrupt sources:

- **External Request Interrupt (EXT*x*)** is an interrupt to the core processor that is asserted by the QUICC Engine module while it is processing an external request. This mechanism allows for the QUICC Engine RISC to interrupt the host core processor due to certain events which occur while the QUICC Engine RISC processes an external request. Up to four external requests are supported by the QUICC Engine module. See Section 19.5, “QUICC Engine External Requests” in Chapter 19, “Configuration.”
- **Virtual Task (VT) Interrupt** is asserted by the RISC when a certain event occur while processing a virtual task. A virtual task is a task that is not directly associated with a peripheral, for example, a UCC. Up to 28 virtual tasks can run concurrently in the QUICC Engine module. The status bit for the virtual task that is issuing the interrupt is found in Section 19.3.3, “QUICC Engine Virtual Tasks Event/Mask Register (CEVTER/CEVTMR).” The exact definition of the assertion of an interrupt in a virtual task is protocol dependent and is not described in this chapter. The virtual tasks on the QUICC Engine module are used to run multiple threads on the UCC Ethernet controller and the ATM controller. See Section 19.6, “Multi-Threading,” for more details.

18.3.13 QUICC Engine RISC Interrupt Mask Register (CRIMR)

CRIMR, shown in Figure 18-28, is used to mask interrupts that are pending in CRIPNR. Setting a bit in CRIMR masks the corresponding pending interrupt in CRIPNR.

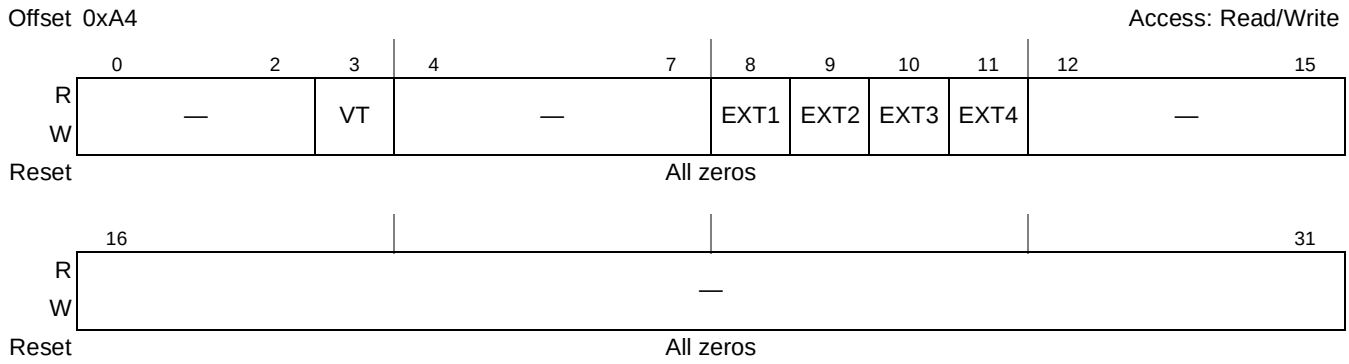


Figure 18-28. CRIMR Fields

18.3.14 QUICC Engine System Interrupt Vector Register (CIVEC)

CIVEC, shown in Figure 18-29, contains a 6-bit vector code (in bits 0–5) representing the unmasked interrupt source of the highest priority level. The ‘Interrupt Vector Code Image’ field (in bits 26-31) is just a duplication of the value in bits 0-5. The vector in this register corresponds to the group of interrupts which are assigned to the QUICC Engine Low interrupt output. See Section 18.2.6, “Interrupt Vector Generation and Calculation,” for the list of codes.

Offset 0x84 Access: Read Only

Bits	0	5	6				25	26	31
R	Interrupt Vector Code		—					Interrupt Vector Code	
W									
Reset	All zeros								

Figure 18-29. QUICC Engine System Interrupt Vector Register (CIVEC)

The CIVEC can be read as either a byte, half word, or a 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 zeroes, allowing indexing into the table, as shown in Figure 18-30.

```
INTR: • • •
        Save State
        R3 <- @ CIVEC
        R4 <-- BASE OF BRANCH TABLE
        • • •
        lbz     RX, R3 (0)    # load as byte
        add     RX, RX, R4
        mtspr   CTR, RX
        bctr
```

BASE	b Routine1
BASE + 4	b Routine2
BASE + 8	b Routine3
BASE + C	b Routine4
BASE +10	•
BASE + n	•

```
INTR: • • •
        Save State
        R3 <- @ CIVEC
        R4 <-- BASE OF BRANCH TABLE
        • • •
        lhz     RX, R3 (0)    # load as half
        add     RX, RX, R4
        mtspr   CTR, RX
        bctr
```

BASE	First Instruction of Routine1
BASE + 400	First Instruction of Routine2
BASE + 800	First Instruction of Routine3
BASE + C00	First Instruction of Routine4
BASE +1000	•
BASE + n	•

Figure 18-30. Interrupt Table Handling Example

CIVEC can be read when an interrupt request occurs. If there are multiple interrupt sources, CIVEC latches the highest priority interrupt. Note that the value of CIVEC cannot change while it is being read.

18.3.15 QUICC Engine High System Interrupt Vector Register (CHIVEC)

CHIVEC, shown in Figure 18-31, contains a 6-bit vector code (in bits 0–5) representing the unmasked interrupt source of the highest priority level. The ‘Interrupt Vector Code Image’ field (in bits 26-31) is just a duplication of the value in bits 0-5. The vector in this register corresponds to the group of interrupts which are assigned to the QUICC Engine High interrupt output. See Section 18.2.6, “Interrupt Vector Generation and Calculation,” for the list of codes.

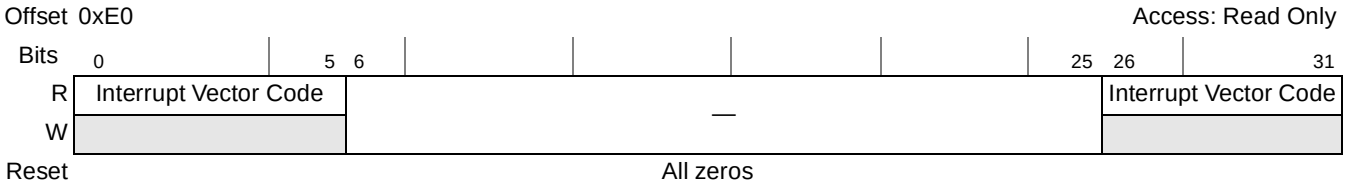


Figure 18-31. QUICC Engine High System Interrupt Vector Register (CHIVEC)

Chapter 19
Configuration

19.1 Introduction

The QUICC Engine control registers allow the core processor to control and monitor the operation of the RISC controllers in the QUICC Engine block. These registers are used to configure certain global options and to create specific commands related to the communication protocols. The commands are issued by writing to the QUICC Engine command register (CECR). These commands are used to initialize the RISC and to control the process of each peripheral controller (UCC, or SPI) while the RISC engine is running.

See Chapter 2, "Memory Map."

19.2 Parameter RAM

The QUICC Engine block maintains a section of the multi-user RAM that contains many parameters for the operation of the communication peripherals (UCCs, SPI). Each peripheral has an associated page in the parameter RAM. The exact definition of the parameter RAM page is contained in each protocol subsection describing a protocol running on a peripheral (such as Ethernet, ATM, and so on.). The size of the parameter RAM page differs from protocol to protocol. The minimum size of the parameter RAM page assigned to any protocol is 64 bytes. The base address of the parameter RAM page must be aligned to 64 bytes.

After reset, the QUICC Engine block initializes the base addresses of the parameter RAM pages for all the protocols to default values which are backward compatible to MPC82xx devices. For MPC832x, the user must override the default values with other values within the multi-user RAM by issuing the ASSIGN PAGE Command to the QUICC Engine block. See Section 19.3.1.1.1, "assign page Command." The advantage of using the ASSIGN PAGE Command is that the user can arrange the parameter RAM area efficiently (without unused areas) according to the specific peripherals/protocols used in the system.

Table 19-1 depicts the default values of the parameter RAM Pages base addresses assigned by the QUICC Engine block to the different peripherals.

Table 19-1. Default Parameter RAM Base Addresses

Address	Peripheral	Size (Bytes)
0x8400	UCC1 (Rx and Tx)	256
0x8500	UCC2 (Rx and Tx)	256
0x8600	UCC3 (Rx and Tx)	256
0x9000	UCC4 (Rx and Tx)	256

Table 19-1. Default Parameter RAM Base Addresses (continued)

Address	Peripheral	Size (Bytes)
0x8000	UCC5 (Rx and Tx)	256
0x8900	SPI1 (Rx and Tx)	128
0x8980	SPI2 (Rx and Tx)	128
0x8A00	TIMER	64
0x8B00	USB (Rx and Tx)	256

As the default values are not in the first 16 Kbytes memory space of the Multi-user RAM, the user has to modify the page addresses using the assign page host command as part of the initialization process. The following table depicts the suggested values of the parameter RAM Pages base addresses for the different peripherals, but any other valid address can be selected.

Table 19-2. QUICC Engine Parameter RAM Base Addresses (Suggested Value for User Configuration)

Address	Peripheral	Size (Bytes)
0x0	UCC1 (Rx & Tx)	256
0x100	UCC2 (Rx & Tx)	256
0x200	UCC3 (Rx & Tx)	256
0x300	UCC4 (Rx & Tx)	256
0x400	UCC5 (Rx & Tx)	256
0x500	SPI1 (Rx & Tx)	128
0x580	SPI2 (Rx & Tx)	128
0x600	TIMER	64
0x700	USB (Rx & Tx)	256

19.3 QUICC Engine Control Registers

19.3.1 QUICC Engine Command Register (CECR)

The CECR register is used by the core processor in order to issue commands to the QUICC Engine block. The commands are actually executed by a QUICC Engine RISC to ensure synchronization with other tasks running on the QUICC Engine block. The core processor sets the CECR[FLG] bit (see Figure 19-1) when it issues a command, and the QUICC Engine block clears the FLG after execution, indicating to the core processor that it is ready for the next command. Subsequent commands to the CECR can be given only after FLG is clear. The software reset command may be issued by setting RST even if the FLG bit is set.

NOTE: On Backward Compatibility

The CECR register corresponds to CPCR register in the CPM. The allocation of the sub block codes allows for backward compatibility with CPM enabled devices according to Table 19-3 below.

Table 19-3. Mapping of CPM FCCs and SCCs to QUICC Engine Block

CPM Peripheral	QUICC Engine Peripheral
FCC1	UCC1
FCC2	UCC2
SCC1	UCC3
SCC3	UCC4
SCC4	UCC5

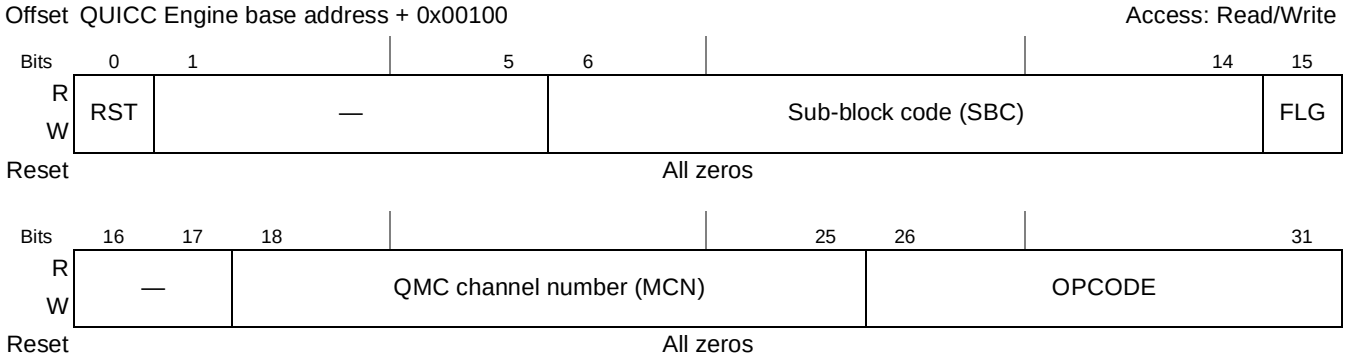


Figure 19-1. QUICC Engine Command Register (CECR)

Table 19-4 describes CECR fields.

Table 19-4. QUICC Engine Command Register Field Descriptions

Bit	Name	Description
0	RST	Software reset command. Set by the core processor and cleared by the QUICC Engine block. It is legal to set RST whether or not FLG is set. The FLG bit is reset as a result of this operation. The user can begin initialization of the QUICC Engine block immediately after this command has completed. RST is useful for resetting the registers and parameters for all the channels (UCCs, SPIs) as well as the QUICC Engine block and RISC timer tables. However, this command does not affect the serial interface (SI*x*) or parallel I/O registers. 0 No software reset 1 Software reset
1–5	—	Reserved

Table 19-4. QUICC Engine Command Register Field Descriptions (continued)

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>6–14</td><td>SBC</td><td>Sub-Block Code. Set by the core processor to specify the sub-block (and its mode) on which the command is to operate.
<table>
<tr><th>SBC</th><th>Sub-block and mode (on which command will operate)</th><th>SBC</th><th>Sub-block and mode (on which command will operate)</th></tr>
<tr><td>100000000</td><td>UCC1
Fast Protocols Mode</td><td>000000000</td><td>UCC1
Slow Protocols Mode</td></tr>
<tr><td>100010000</td><td>UCC2Fast Protocols Mode</td><td>000010000</td><td>UCC2
Slow Protocols Mode</td></tr>
<tr><td>100100000</td><td>UCC3
Fast Protocols Mode</td><td>000100000</td><td>UCC3
Slow Protocols Mode</td></tr>
<tr><td>100110000</td><td>UCC4
Fast Protocols Mode</td><td>000110000</td><td>UCC4
Slow Protocols Mode</td></tr>
<tr><td>101000000</td><td>UCC5
Fast Protocols Mode</td><td>001000000</td><td>UCC5
Slow Protocols Mode</td></tr>
<tr><td>101010000</td><td>Reserved</td><td>001010000</td><td>Reserved</td></tr>
<tr><td>101100000</td><td>Reserved</td><td>001100000</td><td>Reserved</td></tr>
<tr><td>101110000</td><td>Reserved</td><td>001110000</td><td>Reserved</td></tr>
<tr><td>110000000</td><td>Reserved</td><td>010000000</td><td>Reserved</td></tr>
<tr><td>110010000</td><td>USB</td><td>010010000</td><td>Reserved</td></tr>
<tr><td>110100000</td><td>Reserved</td><td>010100000</td><td>SPI1</td></tr>
<tr><td>110110000</td><td>Reserved</td><td>010110000</td><td>SPI2</td></tr>
<tr><td>111000000</td><td>Reserved</td><td>011000000</td><td>Reserved</td></tr>
<tr><td>111010000</td><td>Reserved</td><td>011010000</td><td>Reserved</td></tr>
<tr><td>111100000</td><td>General</td><td>011100000</td><td>Reserved</td></tr>
<tr><td>111110000</td><td>Reserved</td><td>011110000</td><td>Timer</td></tr>
</table></td></tr>
<tr><td>15</td><td>FLG</td><td>Command semaphore flag. Set by the core processor and cleared by the QUICC Engine block.
0 The QUICC Engine block is ready to receive a new command.
1 The CECR contains a command that the QUICC Engine block is currently processing. The QUICC Engine block clears this bit at the end of command execution or after reset.</td></tr>
<tr><td>16–17</td><td>—</td><td>Reserved</td></tr>
<tr><td>18–25</td><td>MCN</td><td>QMC Channel Number. Specifies the channel number in the case of an QMC command.
In UCC protocols, this field contains the protocol code as follows:
0x00 HDLC/transparent
0x0A ATM
0x0C Ethernet
In USB commands (“USB STOP TX” and “USB RESTART TX”) bits 18-23 are reserved and bits 24-25 are the USB End Point Number.</td></tr>
<tr><td>26–31</td><td>OPCODE</td><td>Operation code. Settings are listed in Table 19-5.</td></tr>
</table>

19.3.1.1 QUICC Engine Commands

Table 19-5 shows the QUICC Engine commands in the form of opcodes, and the affect each one has on the QUICC Engine block's peripherals. Be aware that optional microcode packages may contain additional opcode commands not shown here, see the specific microcode documentation for possible additional opcode commands.

As an example of how the opcode commands work, when the opcode is set to 0b00_0000, the UCC*n*, regardless of what mode it is in, performs an Rx and Tx initialization of all parameters, as does the SPI. However, as the table shows, this opcode has no affect on the Timer.

As another example, if the opcode is set to 0b00_0111, and if the UCC is in Slow Protocols Mode, the Receive Buffer Descriptors (RxBDs) are closed for both the UCC and SPI. However, if the UCC is in Fast Protocols Mode, this command is undefined and illegal. In either case, this command is not applicable to the Timer. See Chapter 22, "Unified Communications Controllers (UCCs)," for more information on the UCC: Fast Protocols and Slow Protocols modes.

Table 19-5. QUICC Engine Command Opcodes

Opcode *(command)*	**Affect of Opcode on Indicated Element**			
	UCC*n* protocols: Ethernet, HDLC, ATM (Fast Protocols Mode)	**UCC*n* protocols: UART, BISYNC, HDLC bus, QMC (Slow Protocols Mode)**	**SPI*n***	**Timer**
000000	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	—
000001	INIT RX PARAMS	INIT RX PARAMS	INIT RX PARAMS	—
000010	INIT TX PARAMS	INIT TX PARAMS	INIT TX PARAMS	—
000011	ENTER HUNT MODE	ENTER HUNT MODE	—	—
000100	STOP TX	STOP TX	—	—
000101	GRACEFUL STOP TX	GRACEFUL STOP TX	—	—
000110	RESTART TX	RESTART TX	—	—
000111	—	—	—	—
001000	SET GROUP ADDRESS	—	—	SET TIMER
001001	INSERT CELL	—	—	—
001010	ATM TRANSMIT COMMAND	RESET BCS	—	—
001011	CELL POOL GET	—	—	—
001100	CELL POOL PUT	QMC STOP TX	—	—
001101	IMA HOST COMMAND	QMC STOP RX	—	—
001110	—	—	—	—
001111	PUSHSCHED COMMAND	PUSHSCHED COMMAND	PUSHSCHED COMMAND	PUSHSCHED COMMAND
010000	Reserved			
010001	ATM MULTITHREAD INIT	—	—	—
010010	ASSIGN PAGE	ASSIGN PAGE	ASSIGN PAGE	ASSIGN PAGE

Table 19-5. QUICC Engine Command Opcodes (continued)

Opcode *(command)*	Affect of Opcode on Indicated Element			
	UCC*n* protocols: Ethernet, HDLC, ATM (Fast Protocols Mode)	**UCC*n* protocols: UART, BISYNC, HDLC bus, QMC (Slow Protocols Mode)**	**SPI*n***	**Timer**
010011	SET LAST RECEIVE REQUEST THRESHOLD	—	—	—
010100	START FLOW CONTROL	—	—	—
010101	STOP FLOW CONTROL	—	—	—
010110	ASSIGN PAGE TO DEVICE	ASSIGN PAGE TO DEVICE	ASSIGN PAGE TO DEVICE	ASSIGN PAGE TO DEVICE
011010	GRACEFUL STOP RECEIVE	—	—	—
011011	RESTART RECEIVE (ETHERNET)	—	—	—
011100–011111	Undefined. Reserved for Use by Freescale-supplied RAM Microcodes.			
100000–101011	Reserved			
101100–111111	Undefined. Reserved for Use by Freescale-supplied RAM Microcodes.			

NOTE

If a reserved command is issued, the QUICC Engine block may enter an unknown state that requires an external reset to recover.

NOTE: On Backward-Compatibility

More opcodes are used for additional microcode packages. For example QMC and SS7 have TBD opcode.

NOTE: On Backward-Compatibility

The random number generator on the QUICC Engine block is not user-accessible.

The commands in Table 19-5 are described in Table 19-6.

Table 19-6. Command Descriptions

Command	Description
INIT RX AND TX PARAMS	Initialize transmit and receive parameters. Initializes the transmit and receive parameters of the peripheral controller. 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 TX command is issued.

Table 19-6. Command Descriptions (continued)

Command	Description
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 TX command is issued and the R-bit is set in the next Transmit Buffer Descriptor (TxBD).
RESTART TX	Restart transmission. Once the STOP TX command has been issued, this command is used to restart transmission at the current BD.
SET TIMER	Set timer. Activates, deactivates, or re configures one of the 16 timers in the RISC timer table. See Section 19.4.5, “set timer Command”.
SET GROUP ADDRESS	Set group address. Sets a bit in the hash table for the Ethernet logical group address recognition function.
SET ENTRY IN HASH LOOKUP TABLE	Add entry in Ethernet External or Internal Hash Lookup Table. This command is used in extended filtering mode. See Section 29.8, “Ethernet Command Set.” The SBC field in CECR register must contain the value 0x1E0 as defined in the entry named ‘General’ in the SBC table.
RESET BCS	Reset block check sequence. Used in BISYNC mode to reset the block check sequence calculation.
ATM TRANSMIT	See Section 30.4.1, “ATM Commands.”
QMC STOP TX	Disable the transmit of data on the selected channel and clear the POL in the CHAMR register.
QMC STOP RX	Force the receiver of the selected channel to stop receiving.
PUSHSCHED	Modify the start address for the task running on a given peripheral
ATM MULTI THREAD INIT	See Section 30.4.1.3, “ATM Multi-Thread Init.”
ASSIGNING PAGE	See Section 19.3.1.1.1, “assign page Command”.
GRACEFUL STOP RECEIVE	See Section 29.8, “Ethernet Command Set”.
RESTART RECEIVE	See Section 29.8, “Ethernet Command Set”.

19.3.1.1.1 ASSIGN PAGE Command

This command is used to modify the default value of the Parameter RAM Base Address for a given Peripheral (UCC, SPI). The user needs to program the Parameter RAM Base Address in the CECDR Register (six least significant bits must be zero), and the SNUM in CECR[7–14] Register. See Section 19.7, “Serial Number (SNUM),” for a description of the term SNUM and the values to be used.

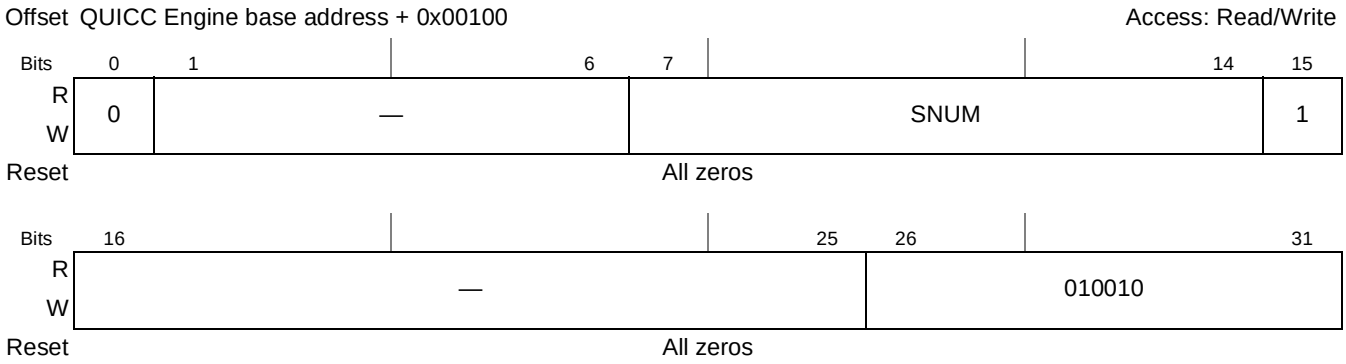


Figure 19-2. CECR for ASSIGN PAGE Command

NOTE

When using the ASSIGN PAGE Command to modify the default value of the Parameter RAM base address associated with the TIMER SNUM, it is mandatory to issue the ASSIGN PAGE Command twice: First assign the TIMER SNUM with the desired base address and then assign the 'Lowest' SNUM (See Table 19-17) to the same base address.

NOTE

When using the ASSIGN PAGE Command to modify the default value of the Parameter RAM base address associated with a peripheral which has Tx and Rx capability, it is mandatory to issue the ASSIGN PAGE Command twice: First assign the Peripheral Rx SNUM with the desired base address and than assign the Peripheral Tx SNUM to the same base address (See Table 19-17). Alternatively it is possible to issue the ASSIGN PAGE TO DEVICE command only once (See Section 19.3.1.1.2, "assign page to Device").

19.3.1.1.2 ASSIGN PAGE to Device

This command is used to modify the default value of the Parameter RAM base address for a given Peripheral (UCC, SPI). The user needs to program the Parameter RAM base address in the CECDR Register (six least significant bits must be zero), and the SBC field in CECR[6–14] Register, corresponding to the peripheral.

19.3.1.1.3 PUSHSCHED Command

This command is used to modify the start address for the task running on a given peripheral, identified by its SNUM. The next time the scheduler selects this peripheral the RISC will start running from the address programmed in this register. Note that if the Peripheral is enabled while giving this command, the old start address may still be used once before the new start address takes effect. Therefore it is not advisable to change this address while a peripheral is enabled.

The new start address is programmed in the CECDR register, the relevant SNUM is programmed in CECR[7–14], and the scheduler optional microcode request is enabled by setting CECR[17].

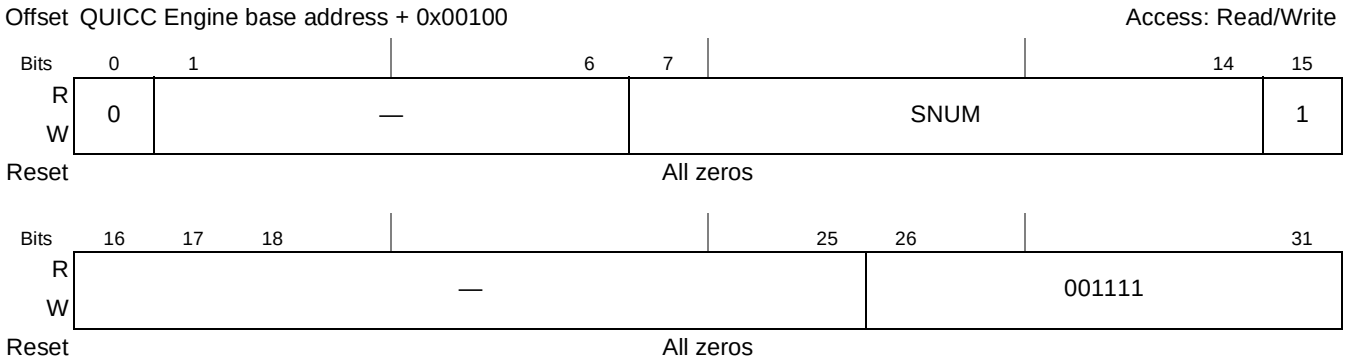


Figure 19-3. CECR for PUSHSCHED Command

19.3.2 QUICC Engine Command Data Register (CECDR)

This register is used to transfer data as a result of the command written to the CECR. If the CECDR data is used by the issued command, the core processor must write the data to the CECDR before setting the CECR[FLG] bit. The CECDR may be updated again only after CECR[FLG] is cleared by the RISC.

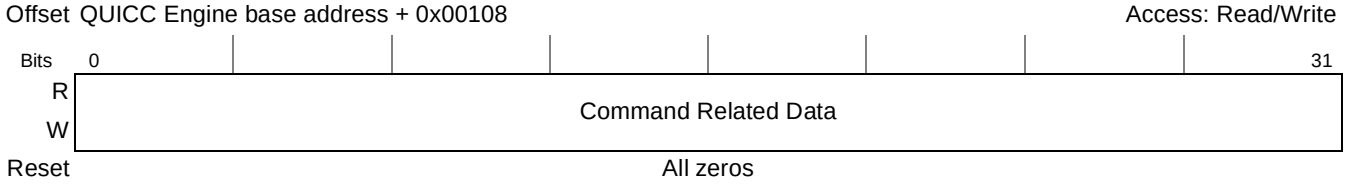


Figure 19-4. QUICC Engine Command Data Register (CECDR)

Refer to specific protocol commands for the contents of this register (for example, Section 29.8.1, “Init Tx, Init Rx and InitTx and Rx Parameters Command”).

19.3.3 QUICC Engine Virtual Tasks Event/Mask Register (CEVTER/CEVTMR)

Usually the QUICC Engine RISC runs tasks associated with some peripheral. However, the QUICC Engine block is capable of running tasks which are not directly associated with a peripheral. See Section 19.6, “Multi-Threading”. The CEVTER is used to report events generated by virtual tasks running on the QUICC Engine block. Each task has one bit associated with it in CEVTER. The CEVTER can be read at any time and bits are cleared only by writing ones, writing zeros does not affect bit values.

The CEVTMR is used to enable interrupts that can be generated in the CEVTER. Setting a CEVTMR bit enables the corresponding interrupt in the CEVTER; clearing a bit masks the corresponding interrupt.

Offset CEVTER: QUICC Engine base address + 0x00130
CEVTMR: QUICC Engine base address + 0x00134

Access: Read/Write

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VT0	VT1	VT2	VT3	VT4	VT5	VT6	VT7	VT8	VT9	VT10	VT11	VT12	VT13	—	
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VT16	VT17	VT18	VT19	VT20	VT21	VT22	VT23	VT24	VT25	VT26	VT27	VT28	VT29	—	
W																
Reset	All zeros															

Figure 19-5. Virtual Task Event/Mask Register (CEVTER/CEVTMR)

Table 19-7 shows the CEVTER field descriptions.

Table 19-7. CEVTER Field Descriptions

Bits	Name	Description
0–13	VT*n*	Virtual Tasks 0–13 0 No event detected 1 Event detected
14–15	—	Reserved
16–29	VT*n*	Virtual Task 16–29 0 No event detected 1 Event detected
30–31	—	Reserved

Table 19-8 shows the CEVTMER field descriptions.

Table 19-8. CEVTMR Field Descriptions

Bits	Name	Description
0–13	VT*n*	Virtual Tasks 0–13 0 Interrupt is masked 1 Interrupt is enabled
14–15	—	Reserved
16–29	VT*n*	Virtual Task 16–29 0 Interrupt is masked 1 Interrupt is enabled
30–31	—	Reserved

19.3.4 QUICC Engine RAM Control Register (CERCR)

This register controls the mode of operation of the internal multi-user RAM (MURAM) and the QUICC Engine instruction RAM (I-RAM). After reset, both RAMs operate in ECC disabled mode. To switch to

ECC enabled mode all RAMs must first be initialized, by writing all memory locations, to put them in a known, cleared state. After the initialization, ECC mode can be enabled. Figure 19-6. shows the register containing the ECC control bits.

Offset QUICC Engine base address + 0x00138 Access: Read/Write

Bits	0	1	2	3	4	5	15
R	MEE	IEE	—		—	—	
W							
Reset	All zeros						

Figure 19-6. QUICC Engine RAM Control Register (CERCR)

Table 19-9. CERCR Field Descriptions

Bits	Name	Description
0	MEE	Multi-user RAM ECC enable 0 ECC disabled in the multi-user RAM. 1 ECC enabled in the multi-user RAM.
1	IEE	Instruction RAM ECC enable 0 ECC disabled in the instruction RAM. 1 ECC enabled in the instruction RAM.
2–4	—	Reserved, should be cleared.
5–15	—	Reserved, should be cleared.

19.3.5 I-RAM Address Register (IADD)

The I-RAM Address Register (IADD) is used to program the address of the instruction RAM of the QUICC Engine block. The data for the address in the IADD register, is written in the IDATA register. Address auto increment can be configured to allow repeated accesses to the IDATA register without a manual update of the IADD register. The initial value of the auto increment counter is programmed in the IADD.

NOTE

Once auto increment is enabled, the IADDR field becomes read-only. To write the IADDR field again, auto increment must first be disabled, IADD[AIE] cleared.

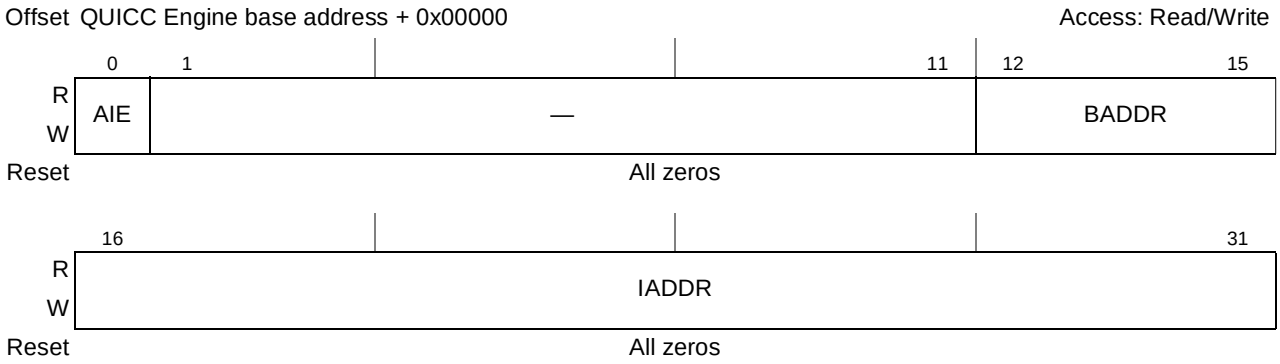


Figure 19-7. IRAM Address Register (IADD)

IADD field descriptions are shown in Table 19-10.

Table 19-10. IADD Field Descriptions

Bits	Name	Description
0	AIE	Auto Increment Enable 0 Auto increment is disabled. 1 Auto increment is enabled. The I-RAM automatically increments the I-RAM address to the next I-RAM entry.
1–11	—	Reserved, should be cleared
12–15	BADDR	Base Address. User should program to 0x8.
16–31	IADDR	I-RAM Address. Bits 30–31 should be cleared (address should be word aligned). This address is updated every read or write, during auto increment (a value of 4 is added with each operation). Valid range for IADD is 0x1FFC.

19.3.6 I-RAM Data Register (IDATA)

The I-RAM Data Register (IDATA) is used to access the I-RAM. An access to the IDATA register will result in the corresponding I-RAM data access according to the current corresponding IADDR content.

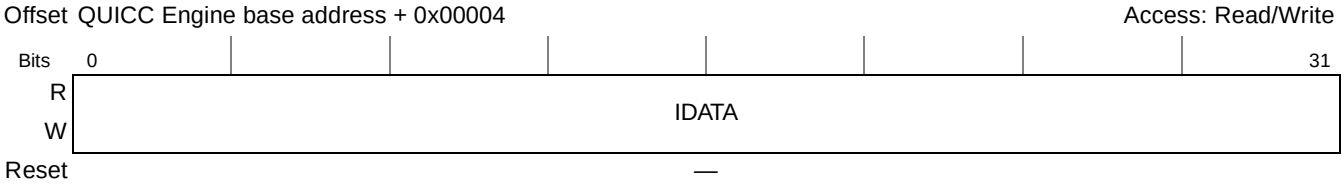


Figure 19-8. IRAM Data Register (IDATA)

Table 19-11. IDATA Field Descriptions

Bits	Name	Description
0–31	IDATA	Data written to or read from the I-RAM at the specified IADD[IADDR] address.

19.3.7 QUICC Engine Microcode Revision Number

The CEURNR register is a 32-bit read-only register which contains the microcode revision number.

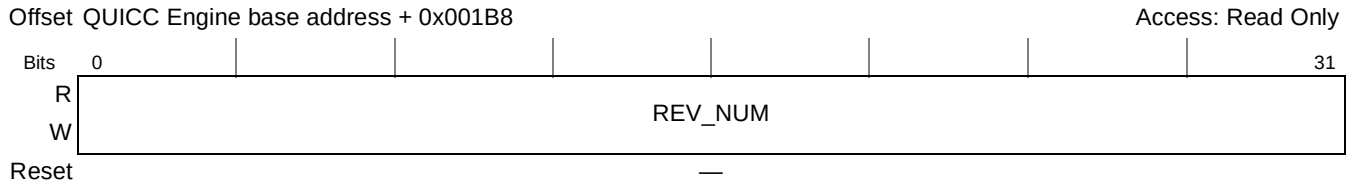


Figure 19-9. QUICC Engine Microcode Revision Number Register (CEURNR)

Table 19-12. CEURNR Field Descriptions

Bits	Name	Description
0–31	REV_NUM	Microcode revision number. 0xCE10_0000

19.3.8 QUICC Engine Controller Configuration Register (CECCR)

The QUICC Engine controller configuration register (CECCR), as shown in Figure 19-10, configures the QUICC Engine block's internal timer (for the QUICC Engine Timer Tables) and the external request modes.

Up to four external requests are connected to the QUICC Engine block. The external requests are used by an external device to request services by the QUICC Engine RISC engine. The request lines may be asserted in level mode or pulse mode and are active high or low as defined in this register. The QUICC Engine block is responsible for turning off the source of the request before exiting the routine handling the request. In level sensitive mode, the external device may leave the external request asserted, and the QUICC Engine block will service the request again.

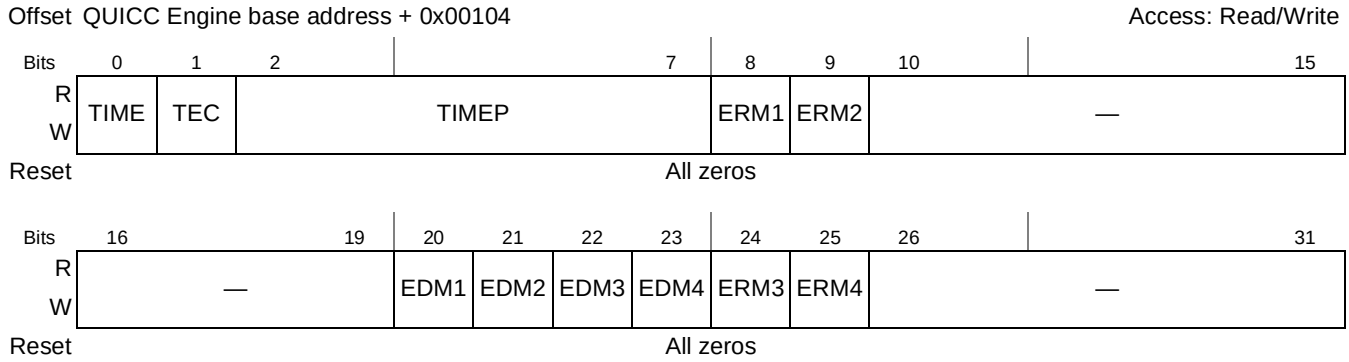


Figure 19-10. QUICC Engine Controller Configuration Register (CECCR)

CECCR bit fields are described in Table 19-13.

Table 19-13. QUICC Engine Controller Configuration Register Field Descriptions

Bits	Name	Description
0	TIME	Timer enable. Enables the QUICC Engine internal timer to generate a trigger to the QUICC Engine block, 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. 0 Disable QUICC Engine internal timer 1 Enable QUICC Engine internal timer
1	TEC	Timer External Clock Enable 0 Use internal QUICC Engine clock to increment the timer. 1 Use external QUICC Engine clock to increment the timer. See Section 20.5.1, “CMX General Clock Route Register (CMXGCR),” on page 20-8. Note on backward Compatibility: In MPC82xx CPM this bit was used to increase the priority of the MCC in the RISC scheduler. Since this functionality is not relevant in QUICC Engine block, this bit is now used for the TEC.
2–7	TIMEP	Timer period controls the RISC timer trigger. The RISC timer tables are scanned upon each timer trigger and the input to the timer trigger generator is the general QUICC Engine clock divided by 1,024. The formula is (TIMEP + 1) × 1,024 = (general QUICC Engine clock period). Thus, a value of 0 stored in these bits gives a timer trigger of 1 × (1,024) = 1,024 general QUICC Engine clocks and a value of 63 (decimal) gives a timer trigger of 64 × (1,024) = 65,536 general QUICC Engine clocks.
8–9	ERM*n*	External Request Mode for *x*= Ext req 1 or 2. Controls the external request pin sensitivity mode. External Request is used to activate the RISC due to an external event. 0 EXT*x* Pin is edge sensitive (according to EDM1) 1 EXT*x* Pin is level sensitive (according to EDM1)
10–19	—	Reserved.
20–23	EDM*n*	External Request Edge Detect Mode for *x*= Ext req 1-4. Controls the external request edge detect mode. 0 EXT*x* Low to High change (if ERM*x*=0) or High level (if ERM*x* = 1) 1 EXT*x* High to Low change (if ERM*x*=0) or Low level (if ERM*x* = 1)
24–25	ERM*n*	External RequestMode for *x*= Ext req 3 or 4. Controls the external request pin sensitivity mode. External Request is used to activate the RISC due to an external event. 0 EXTx Pin is edge sensitive (according to EDM1) 1 EXTx Pin is level sensitive (according to EDM1)
26–31	—	Reserved.

19.3.9 QUICC Engine Time-Stamp Control Register (CETSCR)

The QUICC Engine time-stamp control register (CETSCR), shown in Figure 19-11, configures the QUICC Engine time-stamp timers (CETSR1, CETSR2). The time-stamp timers are used by the ATM and the HDLC controllers. CETSR2 is used by the Ethernet controller.

Offset QUICC Engine base address + 0x0011C Access: Read/Write

Bits	0	1–4	5	6–15
R	EC1	—	RTE1	CETPS1 (Timer Prescale)
W				
Reset	All zeros			

Bits	16	17–20	21	22–31
R	EC2	—	RTE2	CETPS2 (Timer Prescale)
W				
Reset	All zeros			

Figure 19-11. QUICC Engine Time-Stamp Control Register (CETSCR)

Table 19-14 describes CETSCR fields.

Table 19-14. CETSCR Field Descriptions

Bits	Name	Description
0	EC1	External clock 1. Selects the clock used for CETSR1. 0 The time stamp timer is clocked by QUICC Engine clock. 1 The time stamp timer is clocked by a external/brg clock from the clock bank.
1–4	—	Reserved
5	RTE1	Time stamp enable 1. Used to enable CETSR1. 0 Disable time-stamp timer. 1 Enable time-stamp timer.
6–15	CETPS1	Time-stamp timer pre-scale 1. Must be programmed to generate a 1 micro second period input clock to the time-stamp timer. (Time-stamp frequency = (QUICC Engine frequency)/(CETPS1 + 2))
16	EC2	External clock 2. Selects the clock used for CETSR2. 0 The time stamp timer is clocked by QUICC Engine clock. 1 The time stamp timer is clocked by a external/brg clock from the clock bank.
17–20	—	Reserved
21	RTE2	Time stamp enable 2. Used to enable CETSR2. 0 Disable time-stamp timer. 1 Enable time-stamp timer.
22–31	CETPS2	Time-stamp timer pre-scale 2. Must be programmed to generate a 1 micro second period input clock to the time-stamp timer. (Time-stamp frequency = (QUICC Engine frequency)/(CETPS2 + 2))

19.3.10 QUICC Engine Time-Stamp Registers (CETSR*n*)

The QUICC Engine time-stamp registers (CETSR*n*), shown in Figure 19-12, contain the time stamp for time stamp 1 and time stamp 2 respectively.

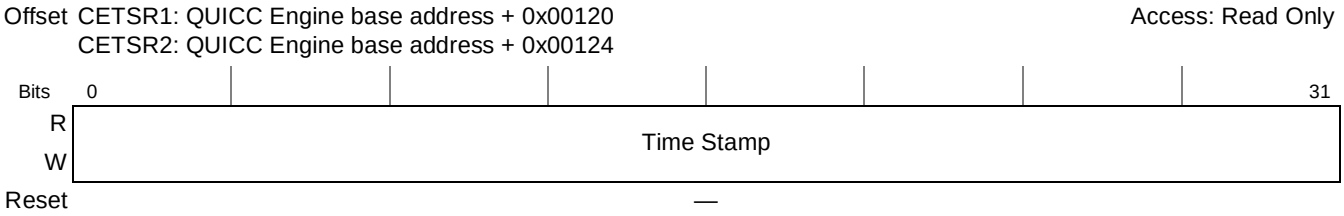


Figure 19-12. QUICC Engine Time-Stamp Registers (CETSR*n*)

After reset, setting CETSCR[RTE1, RTE2] causes the time stamp to start counting microseconds from zero.

19.4 RISC Timer Tables

The QUICC Engine block can control up to 16 software timers that are separate from the four general-purpose timers and the BRGs in the QUICC Engine block. These timers are best used in protocols that do not require extreme precision, but in which it is preferable to free the core processor from scanning the software's timer tables. These timers are clocked from an internal timer that only the QUICC Engine block uses. The following is a list of the RISC timer tables' important features:

- Supports up to 16 timers.
- Two timer modes: one-shot and restart.
- Maskable interrupt on timer expiration.
- Programmable timer resolution as fine as 1024 x Clock_Period (for example, in case a 333 MHz clock is used, timer resolution is as fine as 3.072 micro seconds).
- Continuously updated reference counter.

All operations on the RISC timer tables are based on a fundamental trigger of the QUICC Engine block's internal timer that is programmed in the CECCR, see Section 19.3.8, "QUICC Engine Controller Configuration Register (CECCR)." The trigger value is a multiple of 1,024 internal timer's clocks (see Figure 19-10).

The RISC timer tables have the lowest priority of all QUICC Engine block operations. Therefore, if the QUICC Engine block is so busy with other tasks that it does not have time to service the timer during a trigger interval, one or more timers may not be updated accurately.

The timer table is configured using the CECCR, the timer table parameter RAM, and the QUICC Engine controller timer event/mask registers (see Section 19.4.4, "RISC Timer Event Register (CETER)/Mask Register (CETMR)"), and by issuing SET TIMER to the CECR, see Section 19.3.1, "QUICC Engine Command Register (CECR)".

19.4.1 RISC Timer Table Parameter RAM

Two areas of multi-user RAM, shown in Figure 19-13, are used for the RISC timer tables:

- The RISC timer table parameter RAM
- The RISC timer table entries

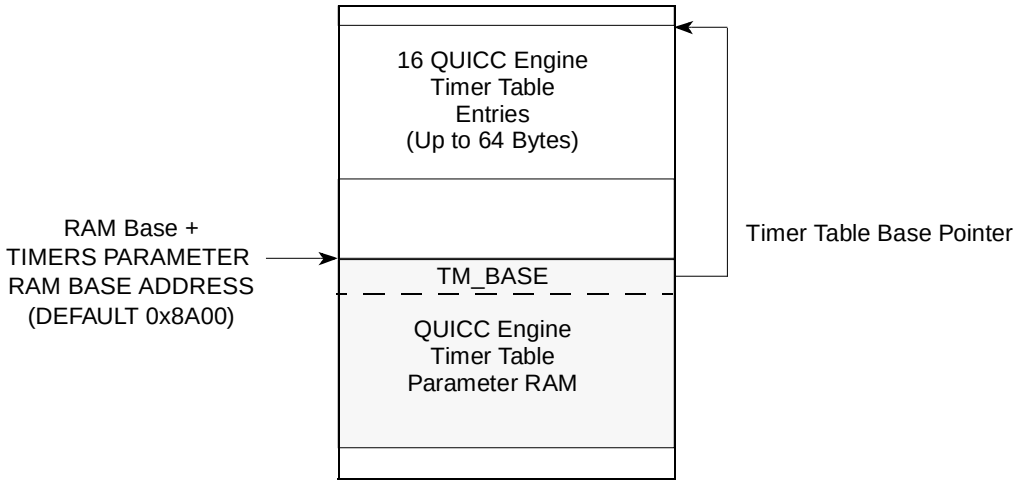


Figure 19-13. RISC Timer Table RAM Usage

The RISC timer table parameter RAM area begins at the RISC timer base address and is used for the general timer parameters; see Table 19-15.

Table 19-15. RISC Timer Table Parameter RAM

Offset[1]	Bits	Name	Description
0x00	0–15	TM_BASE	RISC timer table base address. The actual timers are a small block of memory in the multi-user RAM. TM_BASE is the offset from the beginning of the multi-user 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.
0x02	0–15	TM_PTR	RISC timer table pointer. This value is used exclusively by the QUICC Engine block to point to the next timer accessed in the timer table. It should not be modified by the user.
0x04	0–15	R_TMR	RISC timer mode register. This value is used exclusively by the QUICC Engine block 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	0–15	R_TMV	RISC timer valid register. Used exclusively by the QUICC Engine block to determine if a timer is currently enabled. If the corresponding timer is enabled, a bit is 1. The user should clear this field before activation of the timers. Following initial clear, R_TMV should not be modified by the user, the SET TIMER command should be used instead.

Table 19-15. RISC Timer Table Parameter RAM (continued)

Offset[1]	Bits	Name	Description
0x08	0–31	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 19.4.2, "RISC Timer Command Register (TM_CMD)."
0x0C	0–31	TM_CNT	RISC timer internal count. A trigger counter that the Engine block updates after each trigger. The update occurs after the QUICC Engine block completes scanning the timer table. All 16 timers are scanned every trigger interval regardless of whether any of them is enabled. It is updated if the QUICC Engine block'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 triggers the QUICC Engine block receives and responds to. TM_CNT is updated only after the last timer (timer 15) has been serviced. If the QUICC Engine block is so busy with other tasks that it does not have time to service all the timers during a trigger interval, and timer 15 has not been serviced, then TM_CNT would not be updated in that trigger interval.

1 Offset from timer base address (0x8AE0).

19.4.2 RISC Timer Command Register (TM_CMD)

Figure 19-14 shows the RISC timer command register (TM_CMD).

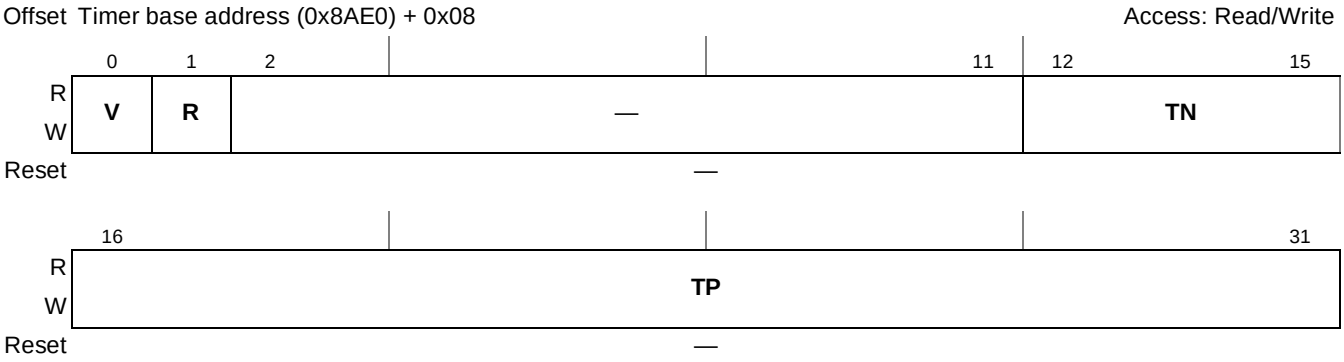


Figure 19-14. RISC Timer Command Register (TM_CMD)

TM_CMD fields are described in Table 19-16.

Table 19-16. TM_CMD Field Descriptions

Bits	Name	Description
0	V	Valid. This bit should be set to enable the timer and cleared to disable it. 0 Timer is disabled 1 Timer is enabled
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 (decimal) 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 all zeroes to the timer period.The maximum value of the timer is 65,536 and is programmed by writing 0xFFFF.

19.4.3 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.

19.4.4 RISC Timer Event Register (CETER)/Mask Register (CETMR)

The CETER is used to report events recognized by the 16 timers and to generate interrupts. CETER can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values.

The RISC timer mask register (CETMR) is used to enable interrupts that can be generated in the CETER. Setting a CETMR bit enables the corresponding interrupt in the CETER; clearing a bit masks the corresponding interrupt. An interrupt is generated only if the RISC timer table bit CIMR[RTT] is set, see Section 18.3.11, “QUICC Engine System Interrupt Mask Register (CIMR).” In the case of Ethernet, bits 1 and 0 can be used to generate an interrupt if an ADD/REMOVE entry in the hash lookup table command fails.

Offset CETER: QUICC Engine base address + 0x00116
CETMR: QUICC Engine base address + 0x0011A

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TMR 15	TMR 14	TMR 13	TMR 12	TMR 11	TMR 10	TMR9	TMR8	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
Reset	All zeros															

Figure 19-15. RISC Timer Event Register/Mask Register (CETER/CETMR)

19.4.5 SET TIMER Command

The SET TIMER command is used to enable, disable, and configure the 16 timers in the RISC timer table. This command is issued by writing the value 0x01E1_0008 to the CECR register. However, before writing this value, the user should program the TM_CMD fields. See Section 19.4.2, “RISC Timer Command Register (TM_CMD).”

19.4.6 RISC Timer Interrupt Handling

The following sequence describes what normally would occur within an interrupt handler for the RISC timer tables:

1. If CIPNR[RTT] is set, indicating an RTT interrupt, then read CETER to see which timers have caused interrupts. The RISC timer event bits can be cleared at 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 RISC timer event bits (CETER) if they were not cleared before. This will clear the CIPNR[RTT] bit too.
4. Execute the RTE instruction.

19.4.7 RISC Timer Table Scan Algorithm

The QUICC Engine RISC scans the timer table once every trigger. It handles each of the 16 timers in turn and checks for other requests with higher priority to service, before handling the next one. For each valid timer in the table, the QUICC Engine block decrements the count and checks for a time out. If none occurs, the QUICC Engine block moves to the next timer. If a time out occurs, the QUICC Engine block sets the corresponding event bit in CETER. Then the QUICC Engine block checks to see if the timer is to be restarted and if it is, the QUICC Engine block 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 QUICC Engine block updates the TM_CNT value in the RISC timer table parameter RAM and stops working on the timer tables until the next trigger.

If a SET TIMER command is issued, the QUICC Engine block makes the appropriate modifications to the timer table and parameter RAM, but does not scan the timer table until the next trigger of the internal timer. It is important to use the SET TIMER command to properly synchronize timer table modifications to the execution of the QUICC Engine block.

19.5 QUICC Engine External Requests

The QUICC Engine RISC may process external requests, by executing a specific microcode routine associated with a certain request. The external requests may represent an external device requiring service from the QUICC Engine RISC. Up to four external requests are supported. External requests are connected to the device via the Parallel I/O Ports. The CECCR register controls certain aspects of external requests See Section 19.3.8, "QUICC Engine Controller Configuration Register (CECCR)," for more details. Events related to the external requests may cause assertion of an interrupt request to the core processor. These events are managed through the CEEXE*n* and CEEXM*n* registers, see Section 19.5.1 below. The external requests resources may be used by various protocols.

19.5.1 QUICC Engine External Request Event and Mask Registers (CEEXE*n*/CEEXM*n*)

The RISC external request event registers (for external requests 1, 2, 3, 4) are set by the microcode when processing external requests. This register is cleared by writing one by the core processor. The mask register is programmed by the core processor to mask background events.

Offset QUICC Engine base address + offset given — Access: Read/Write

CEEXE1: 0x00160 CEEXM1: 0x00164
CEEXE2: 0x00168 CEEXM2: 0x0016C
CEEXE3: 0x00170 CEEXM3: 0x00174
CEEXE4: 0x00178 CEEXM4: 0x0017C

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EXE0	EXE1	EXE2	EXE3	EXE4	EXE5	EXE6	EXE7	EXE8	EXE9	EXE10	EXE11	EXE12	EXE13	EXE14	EXE15
Reset	All zeros															

Figure 19-16. QUICC Engine External Event and Mask Registers (CEEXE*n*/CEEXM*n*)

19.6 Multi-Threading

The UCC Ethernet Controller and the UCC ATM Controller are able to processes frames or cells at high bit rates. In order to achieve these bit rates the UCC receiver and the UCC transmitter are able to simultaneously process multiple frames/cells at any given time. This is implemented with the multithreading mechanism. Each thread processes a different frame/cell. The multithreading mechanism is used in high bit rate protocols, Ethernet and ATM.

The multithreading processing mechanism is comprised of three components: Distributor, Threads and in some cases Terminator. Figure 19-17 depicts the multithreading architecture at a high level.

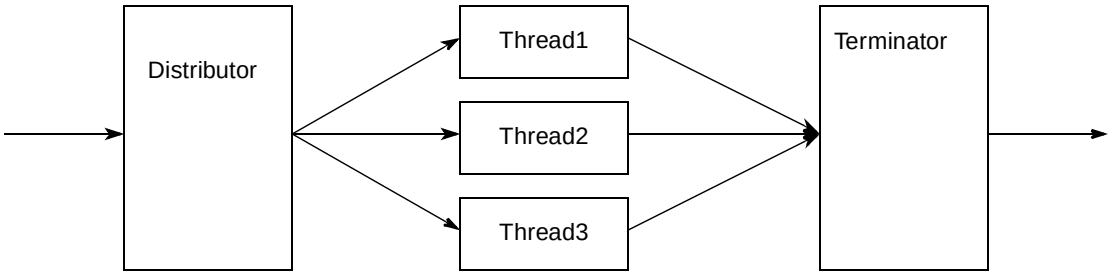


Figure 19-17. Multi-Threading Processing Mechanism

Each one of the components (distributor, threads and terminator) has an ID number associated with it, referred to as its Serial Number (SNUM). The distributor SNUM is always the SNUM of the UCC receiving or transmitting the data.

Each one of the transmitter and receiver threads has its own parameter RAM located in the multi-user RAM.

The user software initializes the values for the SNUM and the pointer of the parameter RAM base address at initialization time. See Section 29.8.1, “Init Tx, Init Rx and InitTx and Rx Parameters Command,” on page 29-107 and Section 30.3.3, “Multi-Threading Structures,” on page 30-63.

Table 19-17 lists the Serial Numbers (SNUMs) to be used to program the multithreading options in the UCCs for Ethernet and ATM.

19.7 Serial Number (SNUM)

Each peripheral has a unique serial number (SNUM) associated with it. Threads, which are used by the multithreading mechanism, see Section 19.6, “Multi-Threading,” have a unique serial number too. In two cases the user should specify the specific SNUM to act upon:

- When issuing the ASSIGN PAGE command, see Section 19.3.1.1.1, “assign page Command”
- When initializing the multithreading mechanism

Table 19-17 shows the unique SNUM for each peripheral and thread. The names under ‘Serial Name’ are Peripheral or Thread names.

Table 19-17. SNUM Table

SNUM	Serial Name	SNUM	Serial Name	SNUM	Serial Name	SNUM	Serial Name
0x00	UCC1 TX	0x40	UCC5 TX	0x80	—	0xC0	—
0x01	UCC1 RX	0x41	UCC5 RX	0x81	—	0xC1	—
0x04	**Thread16**	0x44	—	0x84	—	0xC4	—
0x05	**Thread17**	0x45	—	0x85	—	0xC5	—
0x08	—	0x48	—	0x88	**Thread0**	0xC8	**Thread8**
0x09	—	0x49	—	0x89	**Thread1**	0xC9	**Thread9**
0x0c	**Thread18**	0x4C	—	0x8C	—	0xCC	—
0x0d	**Thread19**	0x4D	—	0x8d	—	0xCD	—
0x10	UCC2 TX	0x50	—	0x90	USB TX	0xD0	—
0x11	UCC2 RX	0x51	—	0x91	USB RX	0xD1	—
0x14	**Thread20**	0x54	—	0x94	—	0xD4	—
0x15	**Thread21**	0x55	—	0x95	—	0xD5	—
0x18	—	0x58	—	0x98	**Thread2**	0xD8	**Thread10**
0x19	—	0x59	—	0x99	**Thread3**	0xD9	**Thread11**
0x1C	**Thread22**	0x5C	—	0x9C	—	0xDC	—
0x1d	**Thread23**	0x5D	—	0x9D	—	0xDD	—
0x20	UCC3 TX	0x60	—	0xA0	SPI1 TX	0xE0	—
0x21	UCC3 RX	0x61	—	0xA1	SPI1 RX	0xE1	—
0x24	**Thread24**	0x64	—	0xA4	—	0xE4	—
0x25	**Thread25**	0x65	—	0xA5	—	0xE5	—
0x28	—	0x68	—	0xA8	**Thread4**	0xE8	**Thread12**
0x29	—	0x69	—	0xA9	**Thread5**	0xE9	**Thread13**
0x2C	**Thread26**	0x6C	—	0xAC	—	0xEC	—
0x2D	**Thread27**	0x6D	—	0xAD	—	0xED	—
0x30	UCC4 TX	0x70	—	0xB0	SPI2 Tx	0xF0	TIMER
0x31	UCC4 RX	0x71	—	0xB1	SPI2 Rx	0xF1	Lowest[1]
0x34	**Thread28**	0x74	—	0xB4	—	0xF4	—
0x35	**Thread29**	0x75	—	0xB5	—	0xF5	—
0x38	—	0x78	—	0xB8	**Thread6**	0xF8	EXT1
0x39	—	0x79	—	0xB9	**Thread7**	0xF9	EXT2
0x3C	EXT3	0x7C	—	0xBC	—	0xFC	—
0x3D	EXT4	0x7D	—	0xBD	—	0xFD	—

[1] See Section 19.3.1.1.1, “assign page Command” for usage of this SNUM.

Chapter 20
Multiplexing and Timers

The QUICC Engine multiplexing and timers logic (CMX) routes clocks and connects the physical interfaces (such as modem lines, TDM lines and proprietary serial lines) to the QUICC Engine peripherals (UCCs, UPC, TDMs, etc.).

The CMX has the following key functions:

- The CMX routes clocks to all the QUICC Engine peripherals from a bank of internal clocks (BRGs 1–11 and 15–16) and a bank of external clocks (1–19) from the parallel I/O ports.
- The CMX connects the UPC internal rate clock to one of several sources as configured by the user.
- The CMX selects which of the UCCs is chosen as serial management interface (SMI) master.
- Per UCC, the CMX works in either NMSI or TDM mode. In NMSI mode, the CMX allows all peripherals to be connected to their own individual pins. In TDM mode, the CMX connects the UCCs to the serial interface. This enables the UCCs to use the time-slot assigner (TSA) which allows the UCCs to multiplex data on the TDM channels. See beginning at Section 20.5.4, "CMX UCC Clock Route Register (CMXUCR1)," for specific details on setting this mode.

NOTE

There is no need to select clocks for UCCs that are routed to the UPC. The clock assignment of the UPC propagates to those UCCs automatically.

Figure 20-1 shows a block diagram of the CMX.

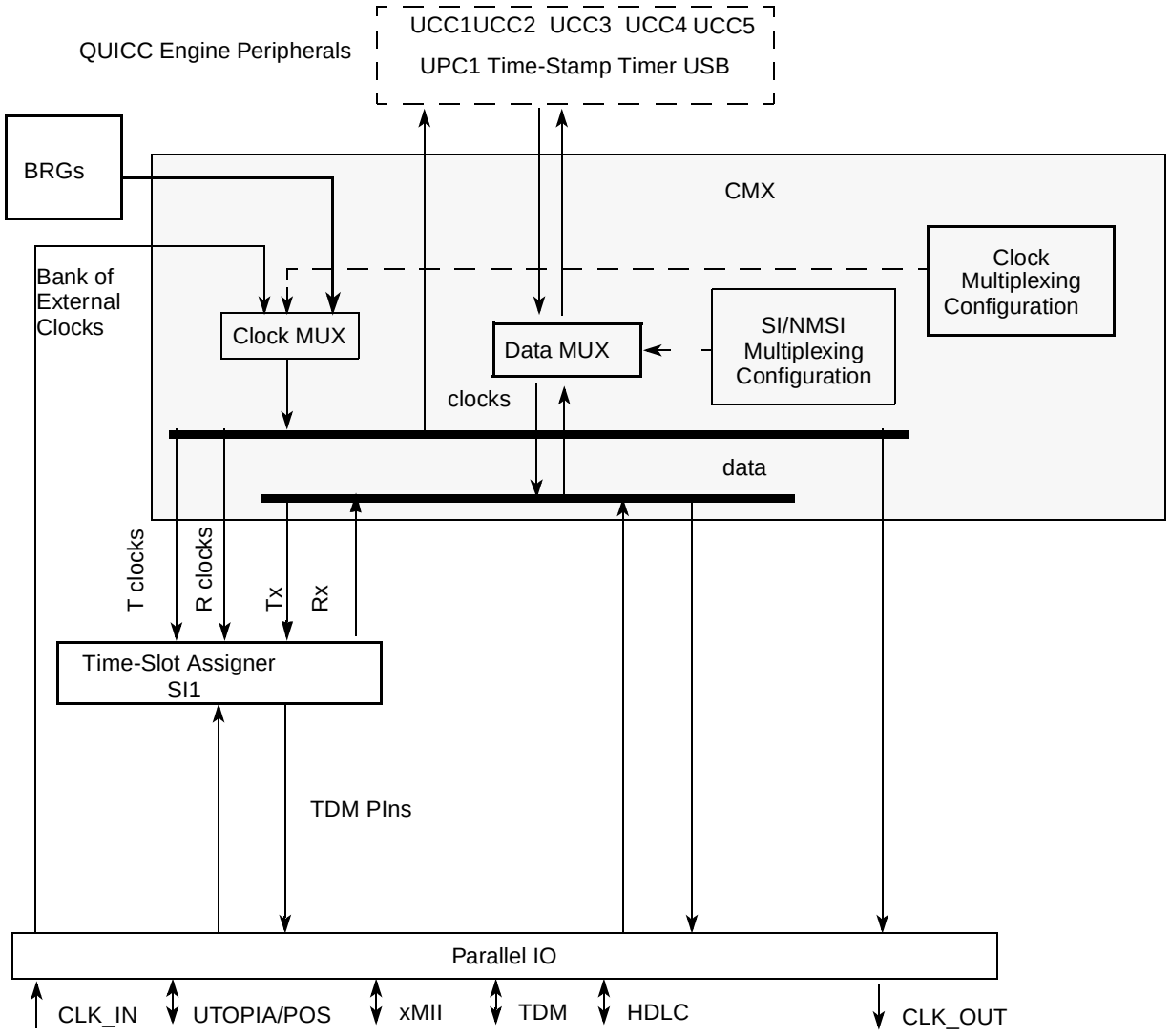


Figure 20-1. QUICC Engine Multiplexing and Timers Logic (CMX) Block Diagram

20.1 Working with Peripherals in NMSI Mode

In NMSI mode, the CMX is primarily used for clock assignment. The clocks can be assigned from an external bank of clock pins or a bank of BRGs, to the following peripherals:

- UCC 1–5
- UPC 1 (Note that UCCs routed to the UPC do not need to be assigned through the CMX)
- Time stamps and Timer.
- USB
- UPC internal rate clock.

- Selection of the serial management interface master from the UCCs.

20.2 Configuring TDM Applications

When working with TDM, the CMX requires clock and sync to be assigned to the SI. See specific details in the CMXSI1CRL, CMXSI1CRH and CMXSI1SYR registers.for SI1 and CMXSI2CRL, CMXSI2CRH and CMXSI2SYR registers for SI2 (SI2 not available on all devices). The clocks and syncs can be assigned from an external bank of nineteen clock pins or a bank of thirteen BRGs to TDM A through D.

See details on the CMXSI registers, as follows:

- Section 20.5.2, "CMX SI1 Clock Route Low Register (CMXSI1CRL)"
- Section 20.5.3, "CMX SI1 SYNC Route Register (CMXSI1SYR)"
- Section 20.5.4, "CMX UCC Clock Route Register (CMXUCR1)"

20.3 Enabling Connections to TSA or NMSI

Each UCC can be independently enabled to connect to the TSA or to dedicated external pins (non-multiplexed serial interface), as shown in Figure 20-2. Once connections are made to the TSA, the clock is routed to UCCs directly from the TSA through the CMX and clock assignment to those UCCs is not required. Exact time slot assignment routing decisions are made in the SI RAM.

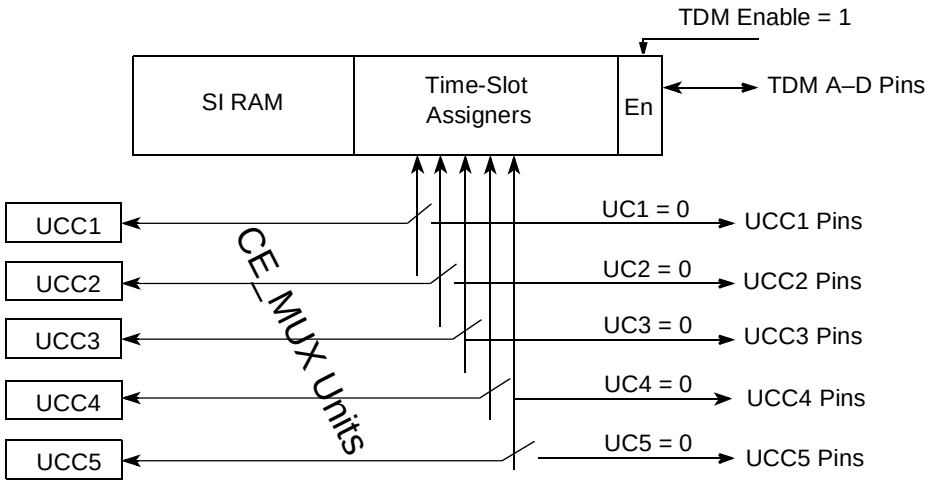


Figure 20-2. Enabling Connections to the TSA

20.4 NMSI Configuration

Only the UCC peripherals can work either with the TDM or directly with their own set of pins in NMSI mode, as configured by the user in CMXUCR*n*[UCy] bits. The connection is configured 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 UCC to connect and where to connect them.

The clocks provided to the peripherals are derived from a bank of internal BRGs and external CLK pins; see Figure 20-3.The clock routing options are described in Table 20-2 and Table 20-2. The bank of clocks selection logic applies an available clock to a peripheral that requires a clock. Because the peripheral is not directly connected to a specific clock source, peripherals can share the same clock.

There are two main advantages to the bank-of-clocks approach. First, a peripheral is not forced to choose a serial device clock from a predefined pin or BRG, providing a flexible pinout-mapping strategy. Second, a group of peripherals 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.

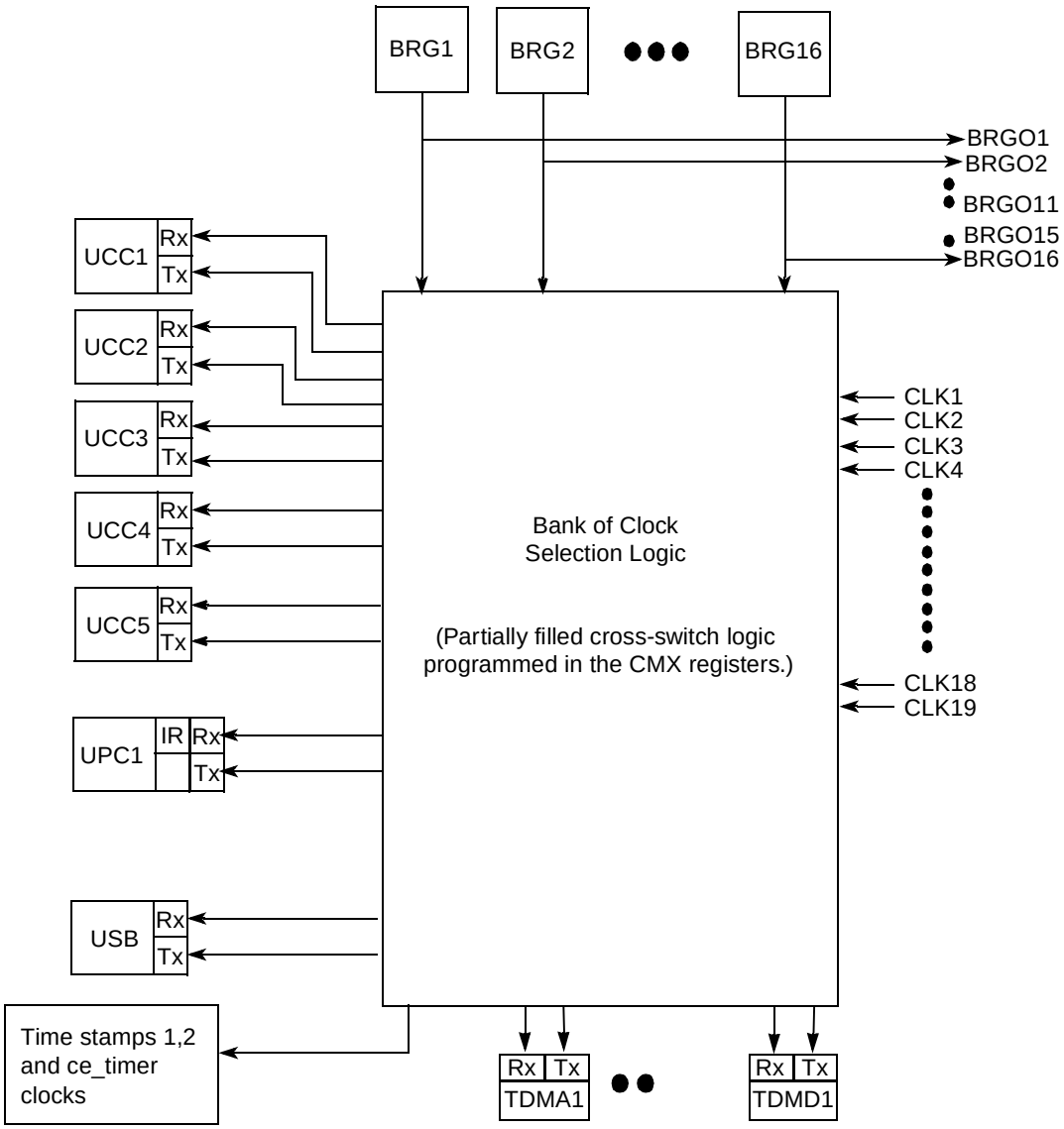


Figure 20-3. Bank of Clocks

The 13 BRGs also make their clocks available to external logic, regardless of whether the BRGs are being used by a serial device. The BRG outputs can be multiplexed with other functions; thus, all BRGO*n* pins may not always be available. The “Signals Description” chapter of the device reference manual shows the function multiplexing.

Consult the tables in this chapter for restrictions in the bank-of-clocks mapping. Table 20-1 shows the external clock source signal options.

Table 20-1. Clock Source Options—External Clock Signals

Clock	External CLK Number																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
UCC1 Rx									V	V	V	V			V	V			
UCC1 Tx									V	V	V	V			V	V			
UCC3 Rx									V	V	V	V			V	V			
UCC3 Tx									V	V	V	V			V	V			
UCC5 Rx													V	V	V	V			V
UCC5 Tx													V	V	V	V			V
UCC2 Rx			V	V			V	V								V	V	V	
UCC2 Tx			V	V			V	V								V	V	V	
UCC4 Rx			V	V			V	V								V	V	V	
UCC4 Tx			V	V			V	V								V	V	V	
UPC1 Rx													V	V	V	V			V
UPC1 Tx													V	V	V	V			V
UPC1 IR																	V	V	V
USB			V		V		V		V				V				V		V
TDMA1 Rx	V	V	V					V											
TDMA1 Tx	V	V		V					V										
TDMB1 Rx	V	V			V					V									
TDMB1 Tx	V	V				V					V								
TDMC1 Rx	V	V					V					V							
TDMC1 Tx	V	V						V					V						
TDMD1 Rx	V	V							V					V					
TDMD1 Tx	V	V								V					V				
Time Stamp 1											V	V							
Time Stamp 2											V	V							
QUICC Engine Timer											V	V							

Table 20-2 shows the internal clock generator options.

Table 20-2. Clock Source Options—Internal Clock Generators

Clock	BRG Clock Number												
	1	2	3	4	5	6	7	8	9	10	11	15	16
UCC1 Rx	V	V					V	V					
UCC1 Tx	V	V					V	V					
UCC3 Rx	V	V					V	V					
UCC3 Tx	V	V					V	V					
UCC5 Rx					V	V	V	V					
UCC5 Tx					V	V	V	V					
UCC2 Rx									V	V		V	V
UCC2 Tx									V	V		V	V
UCC4 Rx									V	V		V	V
UCC4 Tx									V	V		V	V
UPC1 Rx					V	V	V	V					
UPC1 Tx					V	V	V	V					
UPC1 IR			V	V									
USB									V	V			
TDMA1 Rx			V	V									
TDMA1 Tx			V	V									
TDMB1 Rx			V	V									
TDMB1 Tx			V	V									
TDMC1 Rx			V	V									
TDMC1 Tx			V	V									
TDMD1 Rx			V	V									
TDMD1 Tx			V	V									
TDMA1 Rsync									V	V			
TDMA1 Tsync									V	V			
TDMB1 Rsync									V	V			
TDMB1 Tsync									V	V			
TDMC1 Rsync									V		V		
TDMC1 Tsync									V		V		
TDMD1 Rsync									V		V		
TDMD1 Tsync									V		V		
Time Stamp 1											V		
Time Stamp 2											V		
QE Timer											V		

NOTE

After a clock source is selected, the clock is given an internal name. For the UCCs the names are RCLK*n* and TCLK*n*. These internal names specify the clocks sent to the UCCs and are used only in NMSI mode.

Table 20-3 shows the UART autobaud clock options.

Table 20-3. Clock Source Options—UART Autobaud Clock Options

Clock	BRG Clock Number for UART Autobaud												
	1	2	3	4	5	6	7	8	9	10	11	15	16
UCC1 Rx	V												
UCC1 Tx	V												
UCC3 Rx		V											
UCC3 Tx		V											
UCC5 Rx					V								
UCC5 Tx					V								
UCC2 Rx									V				
UCC2 Tx									V				
UCC4 Rx										V			
UCC4 Tx										V			

20.5 CMX Registers

The following sections describe the CMX registers.

Table 20-4. CMX Register Summary

Address[1]	Name	Description
0x400	CMXGCR	Time stamps, SMI, USB
0x404	CMXSI1CRL	TDMA– TDMD RClk and TClk
0x408	—	Reserved
0x40C	CMXSI1SYR	TDMA–TDMD TSYNC and RSYNC
0x410	CMXUCR1	UCC1, UCC3 RxClk and TxClk
0x414	CMXUCR2	UCC5 RxClk and TxClk
0x418	CMXUCR3	UCC2, UCC4 RxClk and TxClk
0x41C	—	Reserved
0x420	CMXUPCR	UPC1 RxClk, TxClk and internal rate clocks

1 Offset from QUICC Engine Base address.

20.5.1 CMX General Clock Route Register (CMXGCR)

The CMX general clock route register (CMXGCR), seen in Figure 20-4, defines the Time stamps, Timer, USB clock sources and the serial management interface master.

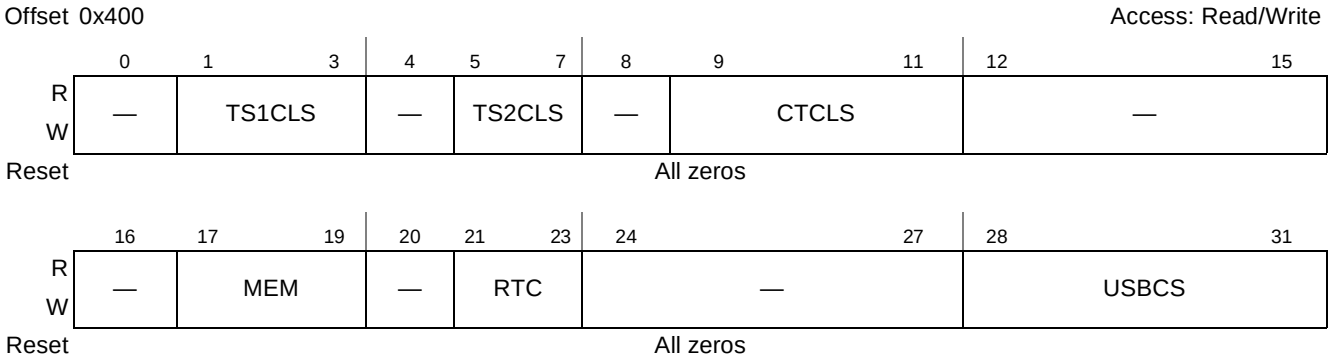


Figure 20-4. CMX General Clock Route Register (CMXGCR)

Table 20-5 describes CMXGCR fields.

Table 20-5. CMXGCR Field Descriptions

Bits	Name	Description
0	—	Reserved. Should be cleared.
1–3	TS1CLS	Time Stamp 1 Clock Source 000 Time Stamp 1 clock source is CLK11 001 Time Stamp 1 clock source is CLK12 010 Reserved 011 Time Stamp 1 clock source is BRG11 100 Time Stamp 1 clock source is External RTC clock
4	—	Reserved. Should be cleared
5–7	TS2CLS	QUICC Engine Time Stamp 2 Clock Source 000 Time Stamp 2 clock source is CLK11 001 Time Stamp 2 clock source is CLK12 010 Reserved 011 Time Stamp 2 clock source is BRG11 100 Time Stamp 2 clock source is External RTC clock
8	—	Reserved. Should be cleared
9–11	CTCLS	QUICC Engine Timer Clock Source 000 QUICC Engine Timer clock source is CLK11 001 QUICC Engine Timer clock source is CLK12 010 Reserved 011 QUICC Engine Timer clock source is BRG11 100 QUICC Engine Timer clock source is External RTC clock
12–16	—	Reserved. Should be cleared

Table 20-5. CMXGCR Field Descriptions (continued)

Bits	Name	Description
17–19	MEM	MII Ethernet management interface select. This field selects the UCC that will be the master of the MII Ethernet management interface. The actual master can be either the selected UCC or the SPI. The selection among the two is made in the QUICC Engine port configuration. 001 UCC2 is the master of the SMI 010 UCC3 is the master of the SMI 011 UCC4 is the master of the SMI
20	—	Reserved. Should be cleared
21–23	RTC	Real Time Clock sources 000 External RTC CLK 001 BRG3 010 BRG4 011 Reserved 100 CLK7 101 CLK8 110 CLK15 111 CLK16
24–27	—	Reserved. Should be cleared.
28–31	USBCS	USB clock source 0000 USB clock is disabled 0001 USB clock is CLK3. 0010 USB clock is CLK5. 0011 USB clock is CLK7. 0100 USB clock is CLK9. 0101 USB clock is CLK13. 0110 USB clock is CLK17. 0111 USB clock is CLK19. 1000 Reserved 1001 USB clock is BRG9. 1010 USB clock is BRG10. 1011–1111 Reserved

NOTE

External clock to the time stamp timer and to the QUICC Engine timer are enabled in CETSCR[EC1,2] and in CECCR[TEC], respectively. See Section 19.3.9, “QUICC Engine Time-Stamp Control Register (CETSCR),” and Section 19.3.8, “QUICC Engine Controller Configuration Register (CECCR).”

20.5.2 CMX SI1 Clock Route Low Register (CMXSI1CRL)

The CMX SI1 clock route low register (CMXSI1CRL), seen in Figure 20-5, defines the connection of SI1 TDM A–D to the clock sources that can be input from the bank of clocks.

Offset 0x404 Access: Read/Write

	0	1–3	4	5–7	8	9–11	12	13–15
R W	—	RTA1CS	—	RTB1CS	—	RTC1CS	—	RTD1CS
Reset	All zeros							

	16	17–19	20	21–23	24	25–27	28	29–31
R W	—	TTA1CS	—	TTB1CS	—	TTC1CS	—	TTD1CS
Reset	All zeros							

Figure 20-5. CMX Si1 Clock Route Low Register (CMXSI1CRL)

Table 20-6 describes CMXSI1CRL fields.

Table 20-6. CMXSI1CRL Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1–3	RTA1CS	Receive TDM A1 clock source 000 TDM A1 receive clock is disabled. 001 TDM A1 receive clock is BRG3. 010 TDM A1 receive clock is BRG4. 011 Reserved 100 TDM A1 receive clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM A1 receive clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 110 TDM A1 receive clock is CLK3. 111 TDM A1 receive clock is CLK8.
4	—	Reserved, should be cleared
5–7	RTB1CS	Receive TDM B1 clock source 000 TDM B1 receive clock is disabled. 001 TDM B1 receive clock is BRG3. 010 TDM B1 receive clock is BRG4. 011 Reserved 100 TDM B1 receive clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM B1 receive clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 110 TDM B1 receive clock is CLK5. 111 TDM B1 receive clock is CLK10.
8	—	Reserved, should be cleared.

Table 20-6. CMXSI1CRL Field Descriptions (continued)

Bits	Name	Description
9–11	RTC1CS	Receive TDM C1 clock source 000 TDM C1 receive clock is disabled. 001 TDM C1 receive clock is BRG3. 010 TDM C1 receive clock is BRG4. 011 Reserved 100 TDM C1 receive clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM C1 receive clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 110 TDM C1 receive clock is CLK7. 111 TDM C1 receive clock is CLK12.
12	—	Reserved, should be cleared
13–15	RTD1CS	Receive TDM D1 clock source 000 TDM D1 receive clock is disabled. 001 TDM D1 receive clock is BRG3. 010 TDM D1 receive clock is BRG4. 011 Reserved 100 TDM D1 receive clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM D1 receive clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 receive clock) 110 TDM D1 receive clock is CLK9. 111 TDM D1 receive clock is CLK14.
16	—	Reserved, should be cleared
17–19	TTA1CS	Transmit TDM A1 clock source 000 TDM A1 transmit clock is disabled. 001 TDM A1 transmit clock is BRG3. 010 TDM A1 transmit clock is BRG4. 011 Reserved 100 TDM A1 transmit clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM A1 transmit clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 110 TDM A1 transmit clock is CLK4. 111 TDM A1 transmit clock is CLK9.
20	—	Reserved, should be cleared
21–23	TTB1CS	Transmit TDM B1 clock source 000 TDM B1 transmit clock is disabled. 001 TDM B1 transmit clock is BRG3. 010 TDM B1 transmit clock is BRG4. 011 Reserved 100 TDM B1 transmit clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM B1 transmit clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 110 TDM B1 transmit clock is CLK6. 111 TDM B1 transmit clock is CLK11.
24	—	Reserved, should be cleared

Table 20-6. CMXSI1CRL Field Descriptions (continued)

Bits	Name	Description
25–27	TTC1CS	Transmit TDM C1 clock source 000 TDM C1 transmit clock is disabled. 001 TDM C1 transmit clock is BRG3. 010 TDM C1 transmit clock is BRG4. 011 Reserved 100 TDM C1 transmit clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM C1 transmit clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 110 TDM C1 transmit clock is CLK8. 111 TDM C1 transmit clock is CLK13.
28	—	Reserved, should be cleared
29–31	TTD1CS	Transmit TDM D1 clock source 000 TDM D1 transmit clock is disabled. 001 TDM D1 transmit clock is BRG3. 010 TDM D1 transmit clock is BRG4. 011 Reserved 100 TDM D1 transmit clock is CLK1. (CLK1 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM D1 transmit clock is CLK2. (CLK2 can be programmed as a common clock for TDM A1,B1,C1,D1 transmit clock) 110 TDM D1 transmit clock is CLK10. 111 TDM D1 transmit clock is CLK15.

20.5.3 CMX SI1 SYNC Route Register (CMXSI1SYR)

The CMX SI1 sync. route register (CMXSI1SYR), seen in Figure 20-6, defines the connection of SI1 to the sync. sources that can be input from the bank of clocks.

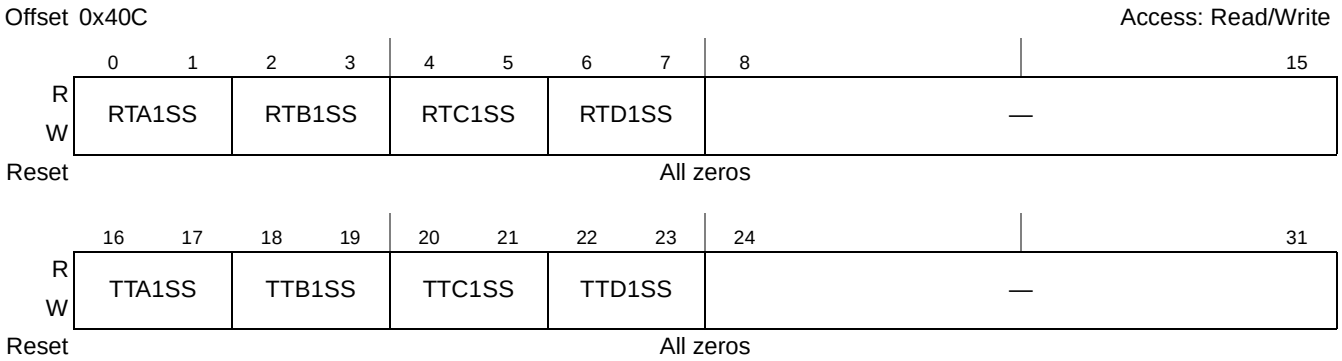


Figure 20-6. CMX SI1 SYNC Route Register (CMXSI1SYR)

Table 20-7 describes CMXSI1SYR fields.

Table 20-7. CMXSI1SYR Field Descriptions

Bits	Name	Description
0–1	RTA1SS	Receive TDM A1 sync. source 00 TDM A1 receive sync is TDM_A1 RSYNC pin 01 TDM A1 receive sync is generated by BRG9. 10 TDM A1 receive sync is generated by BRG10. 11 Reserved
2–3	RTB1SS	Receive TDM B1 sync. source 00 TDM B1 receive sync is TDM_B1 RSYNC pin 01 TDM B1 receive sync is generated by BRG9. 10 TDM B1 receive sync is generated by BRG10. 11 Reserved
4–5	RTC1SS	Receive TDM C1 sync. source 00 TDM C1 receive sync is TDM_C1 RSYNC pin 01 TDM C1 receive sync is generated by BRG9. 10 TDM C1 receive sync is generated by BRG11. 11 Reserved
6–7	RTD1SS	Receive TDM D1 sync. source 00 TDM D1 receive sync is TDM_D1 RSYNC pin 01 TDM D1 receive sync is generated by BRG9. 10 TDM D1 receive sync is generated by BRG11. 11 Reserved
8–15	—	Reserved
16–17	TTA1SS	Transmit TDM A1 sync. source 00 TDM A1 transmit sync is TDM_A1 TSYNC pin 01 TDM A1 transmit sync is generated by BRG9. 10 TDM A1 transmit sync is generated by BRG10. 11 Reserved
18–19	TTB1SS	Transmit TDM B1 sync. source 00 TDM B1 transmit sync is TDM_B1 TSYNC pin 01 TDM B1 transmit sync is generated by BRG9. 10 TDM B1 transmit sync is generated by BRG10. 11 Reserved
20–21	TTC1SS	Transmit TDM C1 sync. source 00 TDM C1 transmit sync is TDM_C1 TSYNC pin 01 TDM C1 transmit sync is generated by BRG9. 10 TDM C1 transmit sync is generated by BRG11. 11 Reserved
22–23	TTD1SS	Transmit TDM D1 sync. source 00 TDM D1 transmit sync is TDM_D1 TSYNC pin 01 TDM D1 transmit sync is generated by BRG9. 10 TDM D1 transmit sync is generated by BRG11. 11 Reserved
24–31	—	Reserved

20.5.4 CMX UCC Clock Route Register (CMXUCR1)

The CMX UCC clock route register (CMXUCR1), shown in Figure 20-7, defines the connection of the UCC1 and UCC3 to the TSA and to the clock sources from the bank of clocks.

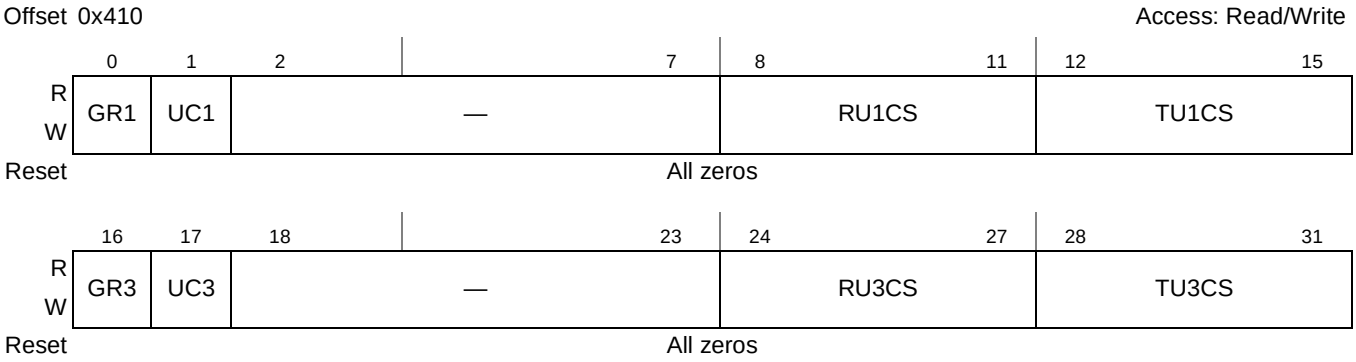


Figure 20-7. CMX UCC Clock Route Register (CMXUCR1)

Table 20-8 describes CMXUCR1 fields.

Table 20-8. CMXUCR1 Field Descriptions

Bits	Name	Description
0	GR1	$\overline{\text{CTS}}$ mode of UCC1 (Valid only in TSA mode) 0 Automatic grant (If CTSP = 0, $\overline{\text{CTS}}$ is always asserted internally, if CTSP=1, $\overline{\text{CTS}}$ is asserted automatically once per TDM frame) 1 IDL grant. The UCC transmitter supports the IDL grant mechanism. Set also the GM*x* bit of the TDM's Mode Register (SI*x*MR1), see Section 32.6.3, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
1	UC1	TSA mode: Defines the UCC1 connection 0 NMSI mode: UCC1 is not connected to the TSA and is either connected directly to the NMSI*x* pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 TSA mode: UCC1 is connected to the TSA of the SIs. The NMSI*x* pins are available for other purposes.
2–7	—	Reserved, should be cleared.
8–11	RU1CS	Receive UCC1 clock source (NMSI mode). Ignored if UCC1 is connected to the TSA (UC1 = 1). 0000 UCC1 receive clock is disabled. 0001 UCC1 receive clock is BRG1. 0010 UCC1 receive clock is BRG2. 0011 UCC1 receive clock is BRG7. 0100 UCC1 receive clock is BRG8. 0101 UCC1 receive clock is CLK9. 0110 UCC1 receive clock is CLK10. 0111 UCC1 receive clock is CLK11. 1000 UCC1 receive clock is CLK12. 1001 UCC1 receive clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5 receive and transmit clocks) 1010 UCC1 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–5 receive and transmit clocks) 1011 UCC1 receive clock is GRX_CLK (use for soft switch from *x*GMII to MII for auto-negotiation) 1100–1111 Reserved

Table 20-8. CMXUCR1 Field Descriptions (continued)

Bits	Name	Description
12–15	TU1CS	Transmit UCC1 clock source (NMSI mode). Ignored if UCC1 is connected to the TSA (UC1 = 1). 0000 UCC1 transmit clock is disabled. 0001 UCC1 transmit clock is BRG1. 0010 UCC1 transmit clock is BRG2. 0011 UCC1 transmit clock is BRG7. 0100 UCC1 transmit clock is BRG8. 0101 UCC1 transmit clock is CLK9. 0110 UCC1 transmit clock is CLK10. 0111 UCC1 transmit clock is CLK11. 1000 UCC1 transmit clock is CLK12. 1001 UCC1 transmit clock is CLK 15 (CLK15 can be programmed as common clock for UCC1–4 receive and transmit clocks). 1010 UCC1 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–4 receive and transmit clocks) 1011 UCC1 transmit clock is TBI RX CLK1 1100–1111 Reserved
16	GR3	$\overline{\text{CTS}}$ mode of UCC3 (Valid only in TSA mode) 0 Automatic grant (If CTSP = 0, $\overline{\text{CTS}}$ is always asserted internally, if CTSP=1, $\overline{\text{CTS}}$ is asserted automatically once per TDM frame) 1 IDL grant. The UCC transmitter support s the IDL grant mechanism. Set also the GM*x* bit of the TDM's Mode Register (SI*x*MR1), see Section 32.6.3, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
17	UC3	Defines the UCC3 connection. 0 UCC3 is not connected to the TSA and is either connected directly to the NMSI*x* pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC3 is connected to the TSA of the SIs. The NMSI*x* pins are available for other purposes.
18–23	—	Reserved, should be cleared

Table 20-8. CMXUCR1 Field Descriptions (continued)

Bits	Name	Description
24–27	RU3CS	Receive UCC3 clock source (NMSI mode). Ignored if UCC3 is connected to the TSA (UC3 = 1). 0000 UCC3 receive clock is disabled. 0001 UCC3 receive clock is BRG1. 0010 UCC3 receive clock is BRG2. 0011 UCC3 receive clock is BRG7. 0100 UCC3 receive clock is BRG8. 0101 UCC3 receive clock is CLK9. 0110 UCC3 receive clock is CLK10. 0111 UCC3 receive clock is CLK11. 1000 UCC3 receive clock is CLK12. 1001 UCC3 receive clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5 receive and transmit clocks) 1010 UCC3 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–5 receive and transmit clocks) 1011–1111 reserved
28–31	TU3CS	Transmit UCC3 clock source (NMSI mode). Ignored if UCC3 is connected to the TSA (UC3 = 1). 0000 UCC3 transmit clock is disabled. 0001 UCC3 transmit clock is BRG1. 0010 UCC3 transmit clock is BRG2. 0011 UCC3 transmit clock is BRG7. 0100 UCC3 transmit clock is BRG8. 0101 UCC3 transmit clock is CLK9. 0110 UCC3 transmit clock is CLK10. 0111 UCC3 transmit clock is CLK11. 1000 UCC3 transmit clock is CLK12. 1001 UCC3 transmit clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5 receive and transmit clocks) 1010 UCC3 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–5 receive and transmit clocks) 1011–1111 Reserved

20.5.5 CMX UCC Clock Route Register (CMXUCR2)

The CMX UCC clock route register (CMXUCR2), seen in Figure 20-8, defines the connection of UCC5 and UCC7 to the TSA and to the clock sources from the bank of clocks. This register also enables the use of the external grant pin.

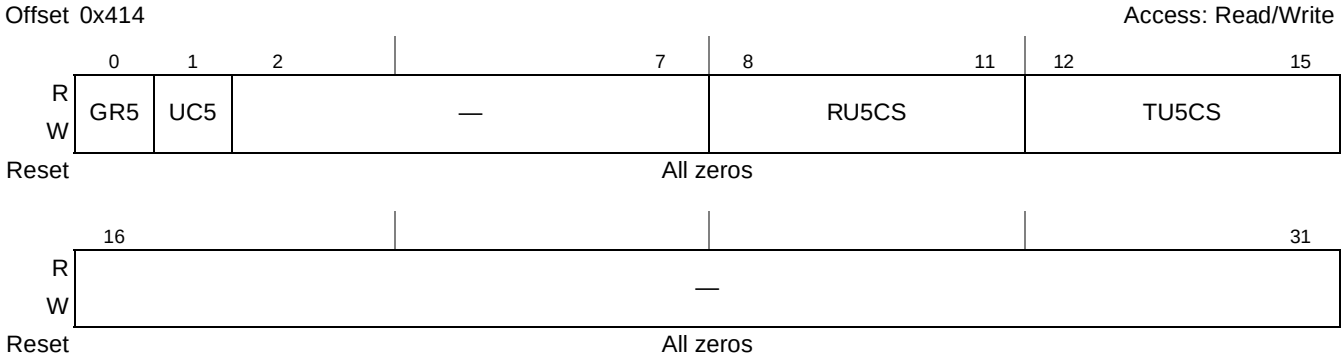


Figure 20-8. CMX UCC Clock Route Register (CMXUCR2)

Table 20-9 describes CMXUCR2 fields.

Table 20-9. CMXUCR2 Field Descriptions

Bits	Name	Description
0	GR5	$\overline{\text{CTS}}$ mode of UCC5 (Valid only in TSA mode) 0 Automatic grant (If CTSP = 0, $\overline{\text{CTS}}$ is always asserted internally, if CTSP=1, $\overline{\text{CTS}}$ is asserted automatically once per TDM frame) 1 IDL grant. The UCC transmitter support the IDL grant mechanism. Set also the GM*x* bit of the TDM's Mode Register (SI*x*MR1), see Section 32.6.3, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
1	UC5	Defines the UCC5 connection 0 UCC5 is not connected to the TSA and is either connected directly to the NMSI*x* pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC5 is connected to the TSA of the SIs. The NMSI*x* pins are available for other purposes.
2–7	—	Reserved, should be cleared
8–11	RU5CS	Receive UCC5 clock source (NMSI mode). Ignored if UCC5 is connected to the TSA (UC5 = 1). 0000 UCC5 receive clock is disabled. 0001 UCC5 receive clock is BRG5. 0010 UCC5 receive clock is BRG6. 0011 UCC5 receive clock is BRG7. 0100 UCC5 receive clock is BRG8. 0101 UCC5 receive clock is CLK13. 0110 UCC5 receive clock is CLK14. 0111 UCC5 receive clock is CLK19. 1000 Reserved 1001 UCC5 receive clock is CLK 15 (CLK15 can be programmed as a common clock for UCC1,3,5,7 receive and transmit clocks) 1010 UCC5 receive clock is CLK 16 (CLK16 can be programmed as a common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved

Table 20-9. CMXUCR2 Field Descriptions (continued)

Bits	Name	Description
12–15	TU5CS	Transmit UCC5 clock source (NMSI mode). Ignored if UCC5 is connected to the TSA (UC5 = 1). 0000 UCC5 transmit clock is disabled. 0001 UCC5 transmit clock is BRG5. 0010 UCC5 transmit clock is BRG6. 0011 UCC5 transmit clock is BRG7. 0100 UCC5 transmit clock is BRG8. 0101 UCC5 transmit clock is CLK13. 0110 UCC5 transmit clock is CLK14. 0111 UCC5 transmit clock is CLK19. 1000 Reserved 1001 UCC5 transmit clock is CLK 15 (CLK15 can be programmed as a common clock for UCC1,3,5,7 receive and transmit clocks) 1010 UCC5 transmit clock is CLK 16 (CLK16 can be programmed as a common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved
16–31	—	Reserved

20.5.6 CMX UCC Clock Route Register (CMXUCR3)

The CMX UCC clock route register (CMXUCR3), shown in Figure 20-9, defines the connection of the UCC2 and UCC4 to the TSA and to the clock sources from the bank of clocks.

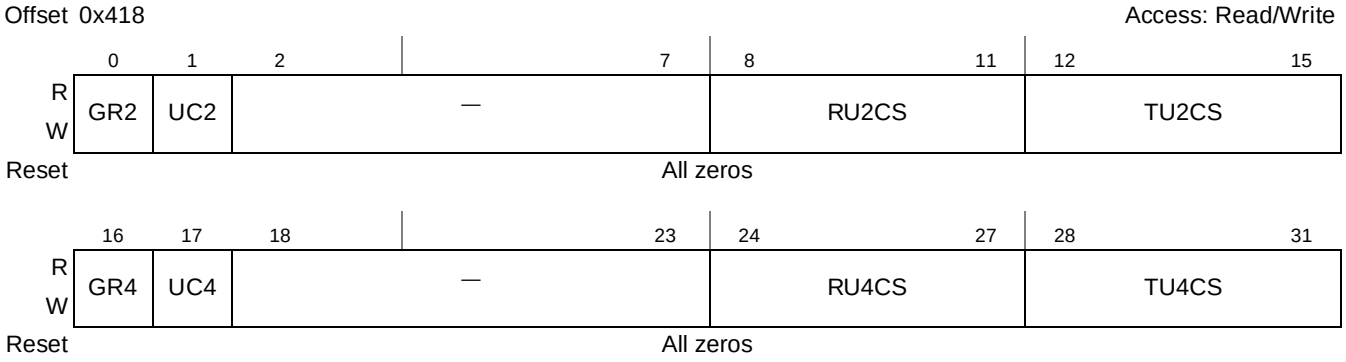


Figure 20-9. CMX UCC Clock Route Register (CMXUCR3)

Table 20-10 describes CMXUCR3 fields.

Table 20-10. CMXUCR3 Field Descriptions

Bits	Name	Description
0	GR2	$\overline{\text{CTS}}$ mode of UCC2 (Valid only in TSA mode) 0 Automatic grant (If CTSP = 0, $\overline{\text{CTS}}$ is always asserted internally, if CTSP=1, $\overline{\text{CTS}}$ is asserted automatically once per TDM frame) 1 IDL grant. The UCC transmitter support the IDL grant mechanism. Set also the GM*x* bit of the TDM's Mode Register (SI*x*MR1), see Section 32.6.3, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
1	UC2	Defines the UCC2 connection 0 UCC2 is not connected to the TSA and is either connected directly to the NMSI*x* pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC2 is connected to the TSA of the SIs. The NMSI*x* pins are available for other purposes.
2–7	—	Reserved, should be cleared
8–11	RU2CS	Receive UCC2 clock source (NMSI mode). Ignored if UCC2 is connected to the TSA (UC2 = 1). 0000 UCC2 receive clock is disabled. 0001 UCC2 receive clock is BRG9. 0010 UCC2 receive clock is BRG10. 0011 UCC2 receive clock is BRG15. 0100 UCC2 receive clock is BRG16. 0101 UCC2 receive clock is CLK3. 0110 UCC2 receive clock is CLK4. 0111 UCC2 receive clock is CLK17. 1000 UCC2 receive clock is CLK18. 1001 UCC2 receive clock is CLK 7 (CLK7 can be programmed as a common clock for UCC2,4,6,8 receive and transmit clocks) 1010 UCC2 receive clock is CLK 8 (CLK8 can be programmed as a common clock for UCC2,4,6,8 receive and transmit clocks) 1011 UCC2 receive clock is CLK 16 (CLK16 can be programmed as a common clock for UCC 1–8 receive and transmit clocks) 1100–1111 Reserved
12–15	TU2CS	Transmit UCC2 clock source (NMSI mode). Ignored if UCC2 is connected to the TSA (UC2 = 1). 0000 UCC2 transmit clock is disabled. 0001 UCC2 transmit clock is BRG9. 0010 UCC2 transmit clock is BRG10. 0011 UCC2 transmit clock is BRG15. 0100UCC2 transmit clock is BRG16. 0101 UCC2 transmit clock is CLK3. 0110 UCC2 transmit clock is CLK4. 0111 UCC2 transmit clock is CLK17. 1000 UCC2 transmit clock is CLK18. 1001 UCC2 transmit clock is CLK 7 (CLK7 can be programmed as a common clock for UCC2,4,6,8 transmit and transmit clocks) 1010 UCC2 transmit clock is CLK 8 (CLK8 can be programmed as a common clock for UCC2,4,6,8 transmit and transmit clocks) 1011 UCC2 transmit clock is CLK 16 (CLK16 can be programmed as a common clock for UCC 1–8 receive and transmit clocks) 1101–1111 Reserved

Table 20-10. CMXUCR3 Field Descriptions (continued)

Bits	Name	Description
16	GR4	$\overline{\text{CTS}}$ mode of UCC4 (Valid only in TSA mode) 0 Automatic grant (If CTSP = 0, $\overline{\text{CTS}}$ is always asserted internally, if CTSP=1, $\overline{\text{CTS}}$ is asserted automatically once per TDM frame) 1 IDL grant. The UCC transmitter support the IDL grant mechanism. Set also the GM*x* bit of the TDM's Mode Register (SI*x*MR1), see Section 32.6.3, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
17	UC4	Defines the UCC4 connection. 0 UCC4 is not connected to the TSA and is either connected directly to the NMSI*x* pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC4 is connected to the TSA of the SIs. The NMSI*x* pins are available for other purposes.
18–23	—	Reserved, should be cleared
24–27	RU4CS	Receive UCC4 clock source (NMSI mode). Ignored if UCC4 is connected to the TSA (UC4 = 1). 0000 UCC4 receive clock is disabled. 0001 UCC4 receive clock is BRG9. 0010 UCC4 receive clock is BRG10. 0011 UCC4 receive clock is BRG15. 0100UCC4 receive clock is BRG16. 0101 UCC4 receive clock is CLK3. 0110 UCC4 receive clock is CLK4. 0111 UCC4 receive clock is CLK17. 1000 UCC4 receive clock is CLK18. 1001 UCC4 receive clock is CLK 7 (CLK7 can be programmed as a common clock for UCC2,4,6,8 receive and transmit clocks) 1010 UCC4 receive clock is CLK 8 (CLK8 can be programmed as a common clock for UCC2,4,6,8 receive and transmit clocks) 1011 UCC4 receive clock is CLK 16 (CLK16 can be programmed as a common clock for UCC 1–8 receive and transmit clocks) 1100–1111 Reserved
28–31	TU4CS	Transmit UCC4 clock source (NMSI mode). Ignored if UCC4 is connected to the TSA (UC4 = 1). 0000 UCC4 receive clock is disabled. 0001 UCC4 transmit clock is BRG9. 0010 UCC4 transmit clock is BRG10. 0011 UCC4 transmit clock is BRG15. 0100 UCC4 transmit clock is BRG16. 0101 UCC4 transmit clock is CLK3. 0110 UCC4 transmit clock is CLK4. 0111 UCC4 transmit clock is CLK17. 1000 UCC4 transmit clock is CLK18. 1001 UCC4 transmit clock is CLK 7 (CLK7 can be programmed as a common clock for UCC2,4,6,8 transmit and transmit clocks) 1010 UCC4 transmit clock is CLK 8 (CLK8 can be programmed as a common clock for UCC2,4,6,8 transmit and transmit clocks) 1011 UCC4 transmit clock is CLK 16 (CLK16 can be programmed as a common clock for UCC 1–8 receive and transmit clocks) 1100–1111 Reserved

20.5.7 CMX UPC Clock Route Register (CMXUPCR)

The CMX UPC Internal Rate clock route register (CMXUPCR), seen in Figure 20-10, defines the source for the UPC1 internal rate clock from an option of the UTOPIA/POS bus clock, an external clock or an internal BRG clock. This clock is the source of the UPC's internal rate prescalar dividers.

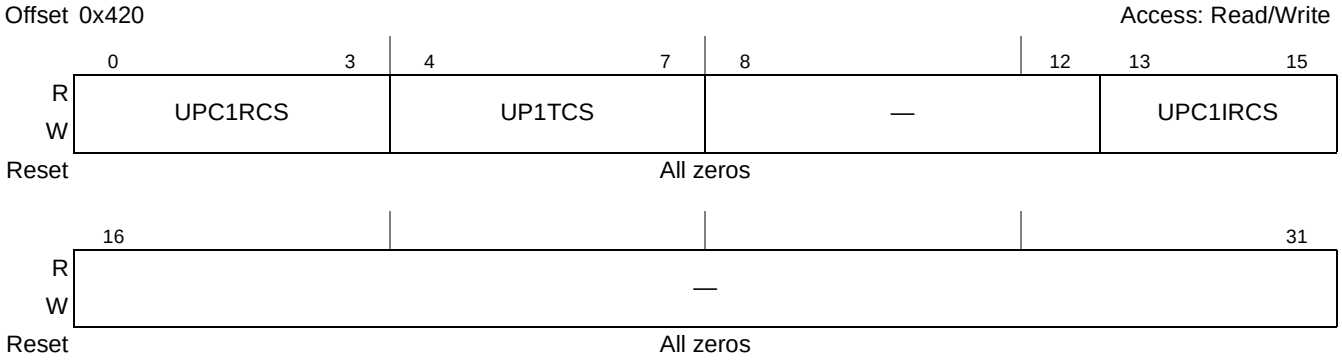


Figure 20-10. CMX UPC Clock Route Register (CMXUPCR)

Table 20-11 describes CMXUPCR fields.

Table 20-11. CMXUPCR Field Descriptions

Bits	Name	Description
0–3	UPC1RCS	UPC1 receive clock source 0000 UPC1 receive clock is disabled 0001 UPC1 receive clock is BRG5 0010 UPC1 receive clock is BRG6 0011 UPC1 receive clock is BRG7 0100 UPC1 receive clock is BRG8 0101 UPC1 receive clock is CLK13 0110 UPC1 receive clock is CLK14 0111 UPC1 receive clock is CLK15 1000 UPC1 receive clock is CLK16 1001 UPC1 receive clock is CLK19 1010 Reserved 1011–1111 Reserved
4–7	UP1TCS	UPC1 transmit clock source 0000 UPC1 transmit clock is disabled 0001 UPC1 transmit clock is BRG5 0010 UPC1 transmit clock is BRG6 0011 UPC1 transmit clock is BRG7 0100 UPC1 transmit clock is BRG8 0101 UPC1 transmit clock is CLK13 0110 UPC1 transmit clock is CLK14 0111 UPC1 transmit clock is CLK15 1000 UPC1 transmit clock is CLK16 1001 UPC1 transmit clock is CLK19 1010 Reserved 1011–1111 Reserved
8–12	—	Reserved. Should be cleared.

Table 20-11. CMXUPCR Field Descriptions (continued)

Bits	Name	Description
13–15	UPC1IRCS	UPC1 Internal Rate clock source 000 UPC1 internal rate clock is the transmit clock of UPC1 001 UPC1 internal rate clock is BRG3 010 UPC1 internal rate clock is BRG4 011 UPC1 internal rate clock is CLK18 100 UPC1 internal rate clock is CLK19 101 UPC1 internal rate clock is CLK17 110–111 Reserved
16–31	—	Reserved. Should be cleared.

20.6 Baud-Rate Generators (BRGs)

The CMX contains 13 independent, identical baud-rate generators (BRGs) that can be used with the UCCs and TDM channels. 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 BRGs' main features:

- Thirteen independent and identical BRGs
- Limited On-the-fly changes allowed (See Section 20.7.1, "BRGC Programming Limitations")
- Each BRG can be routed to one or more UCCs and TDMs
- A 16x divider option allows slow baud rates at high system frequencies
- 8 BRGs contain an autobaud support option
- Each BRG output can be routed to a pin (BRGO*n*)

Figure 20-11 shows the block diagram for a BRG.

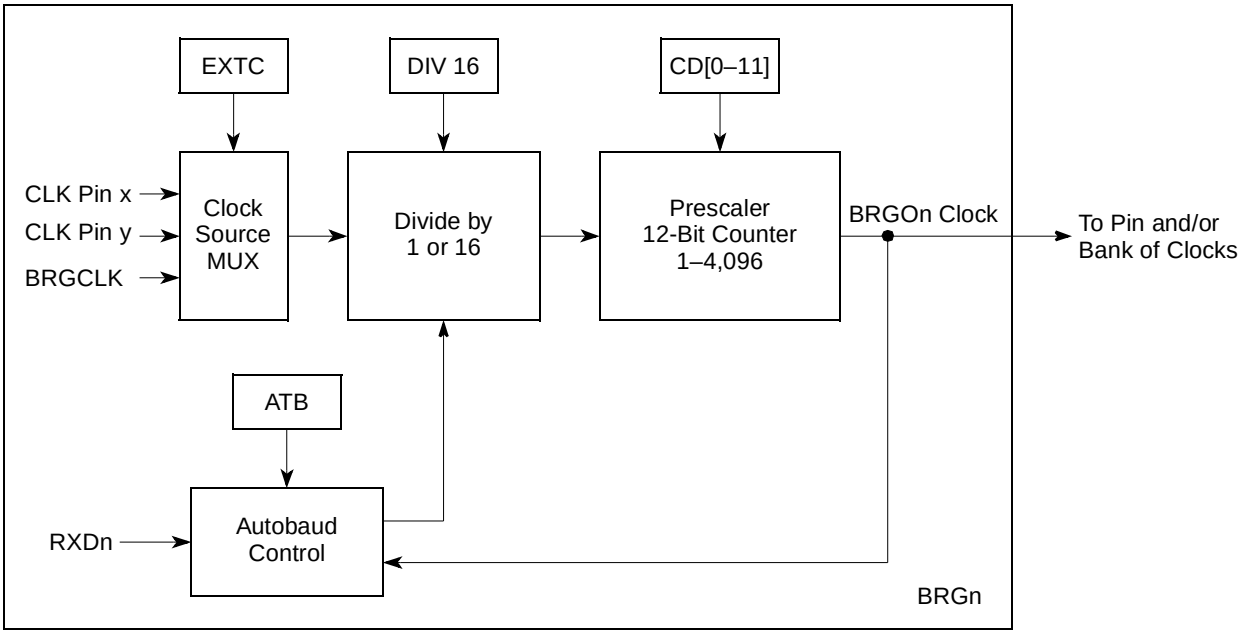


Figure 20-11. Baud-Rate Generator (BRG) Block Diagram

Each BRG clock source can be BRGCLK, or one of two external clocks (selected in BRGC*n*[EXTC]). The BRGCLK is an internal signal generated in the clock synthesizer. Alternatively, external clock pins (CLK Pin x or CLK Pin y) 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 (excluding some specific division ratios – see Section 20.7.1, “BRGC Programming Limitations”), 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 rising edge of the source clock. If the divide factor is odd, the transitions alternate between the falling and rising edges of the source clock. Additionally, the output of the BRG can be sent to the autobaud control block.

20.7 BRG Configuration Registers 1–16 (BRGC*n*)

The BRG configuration registers (BRGC*n*) are shown in Figure 20-12. 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 UCCs 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

(excluding some specific division ratios – see "Section 20.7.1, "BRGC Programming Limitations"), however two changes should not occur within a time equal to two source clock periods.

Offset BRG_BASE+0x00 (BRGC1), BRG_BASE+0x04 (BRGC2), BRG_BASE+0x08 (BRGC3), BRG_BASE+0x0C (BRGC4), BRG_BASE+0x10 (BRGC5), BRG_BASE+0x14 (BRGC6), BRG_BASE+0x18 (BRGC7), BRG_BASE+0x1C (BRGC8), BRG_BASE+0x20 (BRGC9), BRG_BASE+0x24 (BRGC10), BRG_BASE+0x28 (BRGC11), BRG_BASE+0x38 (BRGC15), BRG_BASE+0x3C (BRGC16) Access: Read/Write

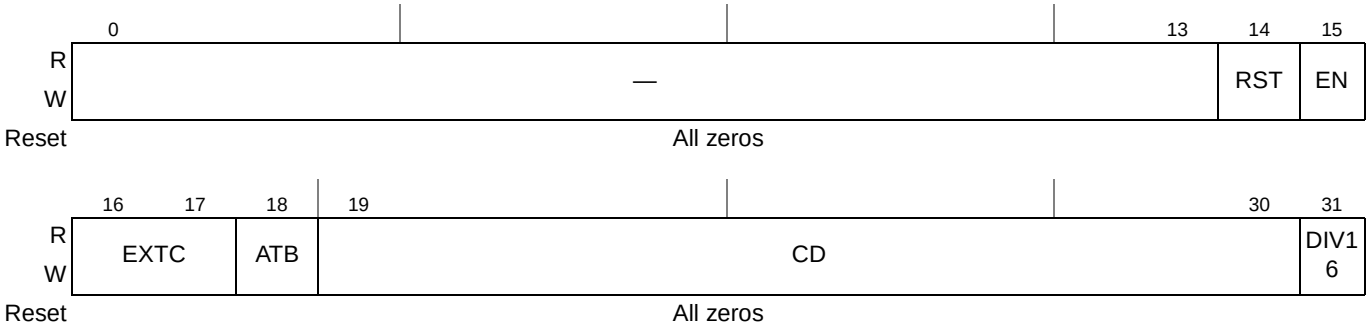


Figure 20-12. Baud-Rate Generator Configuration Registers (BRGC*n*)

Table 20-12 describes the BRGC*n* fields.

Table 20-12. 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.
16–17	EXTC	External clock source. Selects the BRG input clock. See Table 20-13. 00 The BRG input clock comes from the BRGCLK (internal clock generated from the QUICC Engine clock, it is one-half of the QUICC Engine clock); see Section 19.3.8, "QUICC Engine Controller Configuration Register (CECCR)." 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 If BRG9, 10: The BRG input clock comes from the CLK11 pin If BRG11, 15, 16: The BRG input clock comes from the CLK13 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.

Table 20-12. BRGC*n* Field Descriptions (continued)

Bits	Name	Description
18	ATB	Autobaud. Selects autobaud operation of the BRG on the corresponding RXD. ATB must remain zero until the UCC receives the three 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, see Section 24.3.7, "UCC UART Event Register (UCCE) and Mask Register (UCCM)." 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. For more information on the start bit, and this function, see Section 20.8, "Autobaud Operation on a UART for more on the start bit.
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 that the counter has expired and then toggles the clock. See Section 20.9, "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 20.9, "UART Baud Rate Examples." 0 Divide by 1. 1 Divide by 16.

Table 20-13 shows the possible external clock sources for the BRGs.

Table 20-13. 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
BRG1			V		V														
BRG2			V		V														
BRG3									V						V				
BRG4									V						V				
BRG5			V		V														
BRG6			V		V														
BRG7									V						V				
BRG8									V						V				
BRG9											V								
BRG10											V								
BRG11													V						
BRG15													V						
BRG16													V						

20.7.1 BRGC Programming Limitations

There are some guidelines to be kept while programming the BRGC registers:

- On-the-fly changes are allowed, except when changing to or from a CD value of 1, 2 or 3. In these cases, clear EN, set RST and program the new value.
- When moving to or from Autobaud mode, issue the following sequence: clear EN, set RST and program the new value.

20.8 Autobaud Operation on a UART

During the autobaud process, a UART determines 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 upon finding this low level, begins counting the start bit length. During this time, the BRG output clock toggles for 13 BRG clock cycles at the BRG source clock rate and then 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 UCC event register to report that the autobaud controller rewrote BRGC*n*. The interrupt handler can then adjust BRGC*n*[CD, DIV16] (see Table 20-14) 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 UCC associated with this BRG must be programmed to UART mode and select the 16× option for TDCR and RDCR in the general UCC mode register low. Input frequencies such as 1.8432, 3.68, 7.36, and 14.72 MHz should be used. The UCC performing the autobaud function must be connected to that UCC’s BRG; that is, UCC3 must be clocked by BRG3, and so on.

20.8.1 Autobaud Limitations

When using the Autobaud mode, the clock source used by the BRG (internal or external), must be at least 128-times faster then the target divided clock.

An Autobaud operation can happen only once between RSTs. If after an Autobaud operation it is desired to activate it again, issue the following sequence first: clear EN, set RST and program the new value.

20.9 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 66MHz (generated using an external clock source: refer to BRGC*n*[EXTC]), the maximum BRGO rate is 33MHz. Program the UART to 16× oversampling when

using the UCC 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 samples/sec = 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:

$$\text{Async Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \bullet (\text{Clock Divider} + 1) \bullet (\text{Sampling Rate})}$$

$$\text{AsyncBaudRate} = \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \bullet (\text{BRGCx[CD]} + 1) \bullet (\text{GSMRx\_L[xDCR]})}$$

Table 20-14 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 20-14. 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	38,511
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 relationship:

$$\text{Sync Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \bullet (\text{Clock Divider} + 1)}$$

$$\text{SyncBaudRate} = \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \bullet (\text{BRGCx[CD]} + 1)}$$

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.

20.10 Global Timer Module (GTM)

The global timer module (GTM) in the QUICC Engine block includes four 16-bit general-purpose timers, two 32-bit timers or one 64-bit timer. Each GTM timer consists of a timer prescale register (GTPSR), a timer mode register (GTMDR), a timer capture register (GTCPR), a timer counter register (GTCNR), a timer reference register (GTRFR), a timer event register (GTEVR), and a timer global configuration register (GTCFR). The GTPSRs and the GTMDRs contain the primary and secondary prescalers, programmed by the user.

Figure 20-13 shows the functional GTM block diagram.

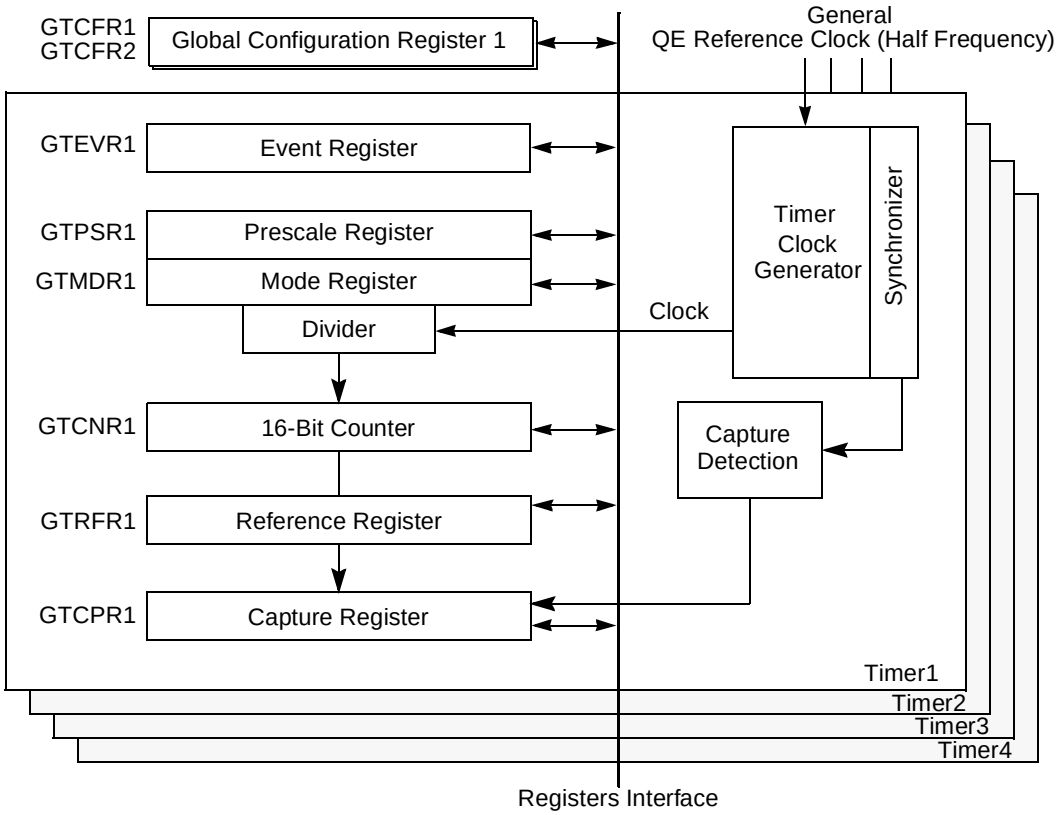


Figure 20-13. Global Timers Block Diagram

20.10.1 GTM Features

The key features of the timer include the following:

- The maximum input clock is the system bus clock
- Four 16-bit programmable timers
- Two timers cascaded internally or externally to form a 32-bit timer
- One timer cascaded internally or externally to form a 64-bit timer

- Maximum period of ~83 msecond (at 200 MHz bus clock) for 16-bit timer
- Maximum period of ~21 second (at 200 MHz bus clock) for 32-bit timer
- Maximum period of thousand of year (at 200 MHz bus clock) for 64-bit timer
- 5-nanosecond timer resolution (at 200 MHz bus clock)
- Two programmable input clock sources for the timer prescalers
- Free run and restart modes
- Functional and programming compatibility with MPC8260 timers

20.10.2 GTM Modes of Operation

The GTM unit can operate in the following modes:

20.10.2.1 GTM Cascaded Modes

GTCFR*n*[PCAS] and GTCFR2[SCAS] are used to put the timers into different cascaded modes:

- Non-cascaded mode: Each timer (timer 1, timer 2, timer 3 and timer 4), function as a independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR and GTCNR. In this mode the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with corresponding 16-bit bus cycles.
- Pair-cascaded mode: In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 may be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer, or two 32-bit timers. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.
- Super-cascaded mode: In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

20.10.2.2 GTM Clock Source Modes

The clock input to the timer’s prescaler can be selected from these sources:

- The QUICC Engine reference clock (one half of the QUICC Engine frequency)
- The system ‘slow go’ clock (QUICC Engine reference clock internally divided by 16)

20.10.2.3 GTM Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding GTMDR selects each mode.

- Free run reference mode (GTMDR*n*[FRR] = 0)

  The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode (GTMDR*n*[FRR] = 1)

  The corresponding timer count is reset immediately after the reference value is reached.

20.10.2.4 GTM Capture Modes

In addition, each timer has a 16-bit field in GTCPR, used to latch the value of the counter when a defined transition of TIN is sensed by the corresponding input capture edge detector.

- Normal gate mode enables the count on a falling edge of the $\overline{\text{TGATE}}$ pin and disables the count on the rising edge of $\overline{\text{TGATE}}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}}$.
- The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of the $\overline{\text{TGATE}}$ pin. This mode has applications in pulse interval measurement and bus monitoring.

20.10.3 GTM External Signals

The QUICC Engine Timers do not have external signals.

20.10.4 GTM Memory Map/Register Definition

The GTM programmable register map occupies 64 bytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect.

All GTM registers are 8 or 16 bits wide, located on 8-bit or 16-bit address boundaries, and should only be accessed as 8-bit or 16-bit quantities. All addresses used in this chapter are offsets from the QUICC Engine Timers base address, as defined in the memory map chapter of your device reference manual. Table 20-15 shows memory map of GTM.

Table 20-15. GTM Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	20.10.4.1/20-31
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	20.10.4.1/20-31
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	20.10.4.2/20-34
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0x0000	20.10.4.3/20-35
0x16	Timer 2 global timers reference register (GTRFR2)			
0x18	Timer 1 global timers capture register (GTCPR1)	R	0x0000	20.10.4.4/20-35
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	20.10.4.5/20-36
0x1E	Timer 2 global timers counter register (GTCNR2)			
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	20.10.4.2/20-34
0x22	Timer 4 global timers mode register (GTMDR4)			

Table 20-15. GTM Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0x0000	20.10.4.3/20-35
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	20.10.4.4/20-35
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	20.10.4.5/20-36
0x2E	Timer 4 global timers counter register (GTCNR4)			
0x30	Timer 1 global timers event register (GTEVR1)	Special	0x0000	20.10.4.6/20-36
0x32	Timer 2 global timers event register (GTEVR2)			
0x34	Timer 3 global timers event register (GTEVR3)			
0x36	Timer 4 global timers event register (GTEVR4)			
0x38	Timer 1 global timers prescale register (GTPSR1)	R/W	0x0003	20.10.4.7/20-37
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			

20.10.4.1 Global Timers Configuration Registers (GTCFR*n*)

GTCFR1 and GTCFR2, shown in Figure 20-14 and Figure 20-15, contain configuration parameters used by the timers. These registers allow simultaneous starting, stopping and resetting of a pair of timers (1 and 2 or 3 and 4) or of a groups of timers (1, 2, 3, and 4) if one bus cycle is used. GTCFR*n* is cleared by reset.

NOTE

For proper operation of the timers, avoid changing the modes of operation and enabling the timer during the same register write operation. The mode can be changed when GTCFR*n*[RST*n*] is cleared, but when GTCFR*n*[RST*n*] is being set, it must be the only bit to change its value.

Offset 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	PCAS	BCM	STP2	RST2	GM2	GM1	STP1	RST1
Reset	All zeros							

Figure 20-14. Global Timers Configuration Register 1 (GTCFR1)

Table 20-16 defines the bit fields of GTCFR1.

Table 20-16. GTCFR1 Field Descriptions

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer. **Note:** This bit will be ignored in Super-cascade Mode (GTCFR2[SCAS]=1). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus the user should first clear the RST1 and RST2 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	BCM	Backward compatible mode 0 Provide backward compatibility to PowerQUICC II family timers. In this mode GTCFR1[GM2] bit will control the gate mode for timers 1 and 2 and GTCFR2[GM4] bit will control the gate mode for timers 3 and 4. GTCFR1[GM1] and GTCFR2[GM3] bits will be ignored. 1 Normal operational mode
2	STP2	Stop timer 2 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 2, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST2	Reset timer 2 0 Reset the timer 2, including GTMDR2, GTRFR2, GTCNR2, GTCPR2 and GTEVR2 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP2 bit is cleared.
4	GM2	Gate mode for $\overline{\text{TGATE2}}$: Should be cleared.
5	GM1	Gate mode for $\overline{\text{TGATE1}}$ Should be cleared.
6	STP1	Stop timer 1 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 1, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	RST1	Reset timer 1 0 Reset the timer 1, including GTMDR1, GTRFR1, GTCNR1, GTCPR1 and GTEVR1 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP1 bit is cleared.

The GTCFR2 register is shown in Figure 20-15.

Offset 0x04 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	PCAS	SCAS	STP4	RST4	GM4	GM3	STP3	RST3
Reset	All zeros							

Figure 20-15. Global Timers Configuration Register 2 (GTCFR2)

Table 20-17 defines the bit fields of GTCFR2.

Table 20-17. GTCFR2 Field Descriptions

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation. 1 Timers 3 and 4 cascade to form a 32-bit timer. **Note:** This bit will be ignored in Super-cascade Mode (GTCFR2[SCAS]=1). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus the user should first clear the RST3 and RST4 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	SCAS	Super cascade mode 0 Normal operation 1 Timers 1, 2, 3 and 4 cascade to form a 64-bit timer. **Note:** In Super-cascade Mode (GTCFR2[SCAS]=1) the Pair-cascade mode bits will be ignored, (GTCFR1/2[PCAS]=Don't Care). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus the user should first clear the RST1, RST2, RST3 and RST4 bits (without changing SCAS) and then, in a separate write to the register, change the value of SCAS.
2	STP4	Stop timer 4 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 4, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST4	Reset timer 4 0 Reset the timer 4, including GTMDR4, GTRFR4, GTCNR4, GTCPR4 and GTEVR4 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP4 bit is cleared.
4	GM4	Gate mode for $\overline{\text{TGATE4}}$ Should be cleared.
5	GM3	Gate mode for $\overline{\text{TGATE3}}$ Should be cleared.
6	STP3	Stop timer 3 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 3, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	RST3	Reset timer 3 0 Reset the timer 3, including GTMDR3, GTRFR3, GTCNR3, GTCPR3 and GTEVR3 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP3 bit is cleared.

20.10.4.2 Global Timers Mode Registers (GTMDR1–GTMDR4)

GTMDR1, GTMDR2, GTMDR3, and GTMDR4 are shown in Figure 20-16. Erratic behavior may occur if GTCFR1 and GTCFR2 are not initialized before the GTMDR*n*. Only GTCFR*n*[RST*n*] and GTCFR*n*[STP*n*] can be modified at any time.

Offset GTMDR1: 0x10
GTMDR2: 0x12
GTMDR3: 0x20
GTMDR4: 0x22

Access: Read/Write

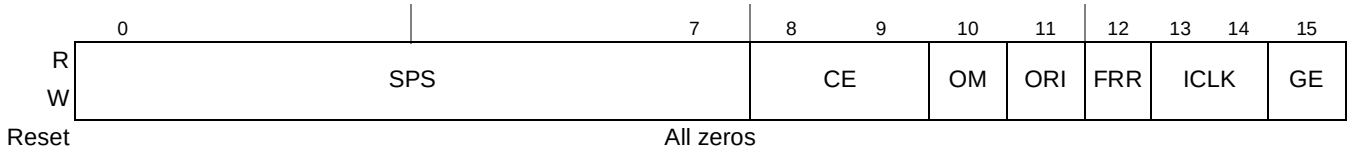


Figure 20-16. Global Timers Mode Registers (GTMDR1–GTMDR4)

Table 20-18 defines the bit fields of GTMDR*n*.

Table 20-18. GTMDR*n* Field Descriptions

Bits	Name	Description
0–7	SPS	Secondary prescaler value The secondary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.
8–9	CE	Capture edge and enable interrupt 00 Disable interrupt on capture event; capture function is disabled All other values are reserved.
10	OM	Output mode: Should be cleared.
11	ORI	Output reference interrupt enable 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt upon reaching the reference value.
12	FRR	Free run/restart mode 0 Free run. The timer count continues to increment after the reference value is reached. 1 Restart. The timer count is reset immediately after the reference value is reached.
13–14	ICLK	Input clock source for the timer. 00 Internally cascaded input. This selection means: For ICLK1, the timer 1 input is the output of timer 2; For ICLK2, the timer 1 input is the output of timer 2, the timer 2 input is the output of timer 3, the timer 3 input is the output of timer 4; For ICLK3, the timer 3 input is the output of timer 4; For ICLK4 this selection means no input clock is provided to the timer. 01 Internal QUICC Engine reference clock. 10 Internal “slow go” clock (divided by 16 QUICC Engine reference clock). 11 Reserved.
15	GE	Gate enable: Should be cleared.

20.10.4.3 Global Timers Reference Registers (GTRFR1–GTRFR4)

GTRFR1, GTRFR2, GTRFR3, and GTRFR4, shown in Figure 20-17 are a 16-bit, memory-mapped, read/write registers containing the 16-bit reference values for the each timer's timeout. GTRFR*n* is set to all ones by reset. The reference value is not reached until GTCNR*n*[CNV] increments to value, equal GTRFR*n*[TRV].

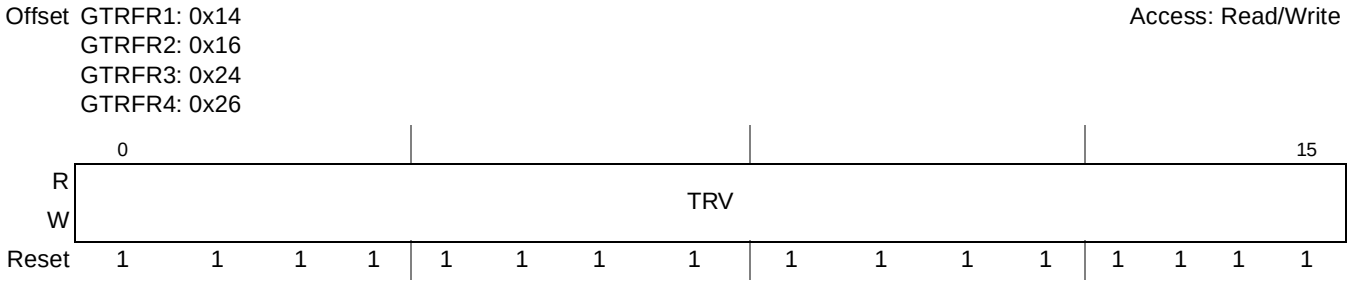


Figure 20-17. Global Timers Reference Registers (GTRFR1–GTRFR4)

Table 20-19 defines the bit fields of GTRFR*n*.

Table 20-19. GTRFR*n* Field Descriptions

Bits	Name	Description
0–15	TRV	Timeout reference value. 16-bit timeout reference value for the corresponding timer. Set to all ones by reset.

20.10.4.4 Global Timers Capture Registers (GTCPR1–GTCPR4)

GTCPR1, GTCPR2, GTCPR3, and GTCPR4, shown in Figure 20-18, are read-only registers.

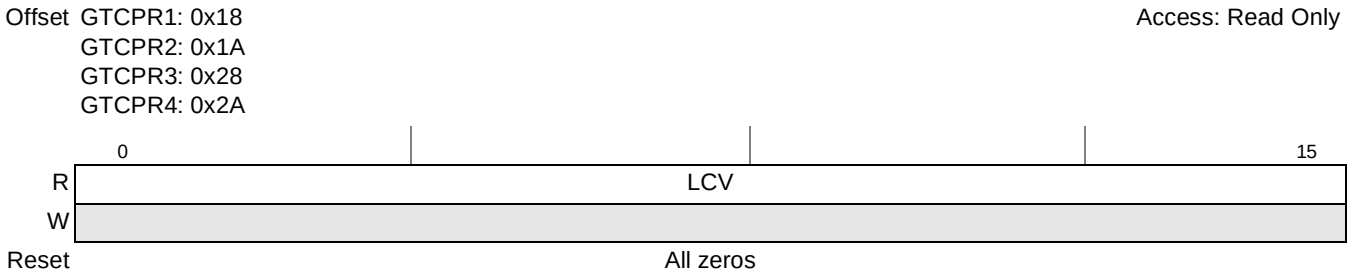


Figure 20-18. Global Timers Capture Registers (GTCPR1—GTCPR4)

Table 20-20 defines the bit fields of GTCPR*n*.

Table 20-20. GTCPR*n* Field Descriptions

Bits	Name	Description
0–15	LCV	Latched counter value Corresponding timer's 16-bit latched value.

20.10.4.5 Global Timers Counter Registers (GTCNR1–GTCNR4)

GTCNR1, GTCNR2, GTCNR3, and GTCNR4, shown in Figure 20-19, are four 16-bit, memory-mapped, read/write up-counters. A read cycle to a GTCNR*n*[CNV] fields yields the current value of the appropriate timer but does not affect the counting operation. A write cycle to a GTCNR*n*[CNV] fields sets the register to the written value, causing its corresponding primary and secondary prescaler counters to be reset.

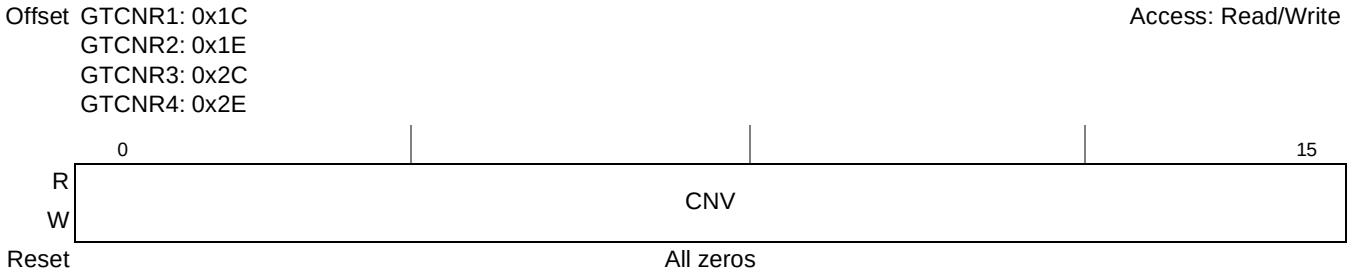


Figure 20-19. Global Timers Counter Registers (GTCNR1–GTCNR4)

Table 20-20 defines the bit fields of GTCNR*n*.

Table 20-21. GTCNR*n* Field Descriptions

Bits	Name	Description
0–15	CNV	Counter value. Corresponding timer's 16-bit read/write up-counter value.

20.10.4.6 Global Timers Event Registers (GTEVR1–GTEVR4)

GTEVR1, GTEVR2, GTEVR3, and GTEVR4, shown in Figure 20-20, are used to report events recognized by any of the timers. On recognition of an output reference event, the appropriate timer sets GTEVR*n*[REF], regardless of the corresponding GTMDR*n*[ORI] bit. The capture event is only set if it is enabled by GTMDR*n*[CE]. GTEVR, which appear to the user as memory-mapped registers, can be read at any time.

Bits are cleared by writing ones to them (writing zeros does not affect bit values). Both bits must be reset before the timer negates the interrupt to the interrupt controller.

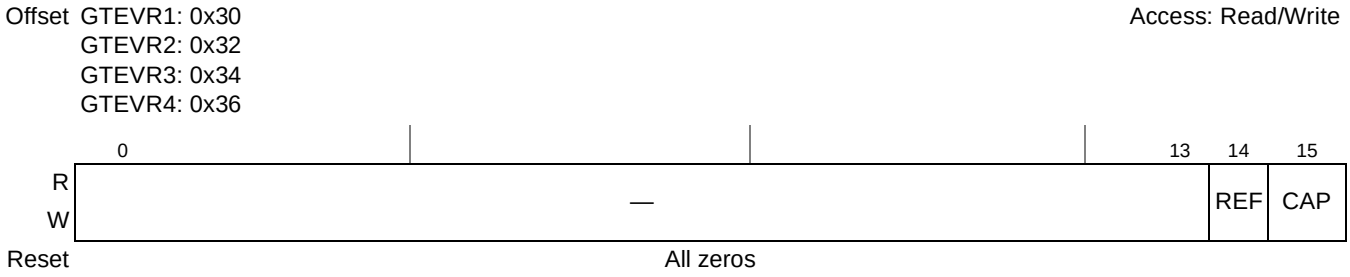


Figure 20-20. Global Timers Event Registers (GTEVR1–GTEVR4)

Table 20-22 defines the bit fields of GTEVR.

Table 20-22. GTEVR Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	REF	Output reference event 0 No event 1 The counter has reached the GTRFR*n*[TRV] value. GTMDR*n*[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Counter capture event: 0 No event 1 Reserved

20.10.4.7 Global Timers Prescale Registers (GTPSR1–GTPSR4)

GTPSR1, GTPSR2, GTPSR3, and GTPSR4 are shown in Figure 20-21. Erratic behavior may occur if GTPSR*n* is not initialized before the corresponding GTMDR*n*. The total timer prescale value is calculated as follows:

$$GTMx_{prescaler} = (GTPSRx[PPS] + 1) \cdot (GTMDRx[SPS] + 1)$$

This gives a total prescale range from 1 (GTPSR*n*[PPS] = 0x00, GTMDR*n*[SPS] = 0x00) to 65,536 (GTPSR*n*[PPS] = 0xFF, GTMDR[SPS] = 0xFF).

Offset GTPSR1: 0x38 — Access: Read/Write
GTPSR2: 0x3A
GTPSR3: 0x3C
GTPSR4: 0x3E

	0							7	8							15
R W	—								PPS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 20-21. Global Timers Prescale Registers (GTPSR1–GTPSR4)

Table 20-23 defines the bit fields of GTPSR.

Table 20-23. GTPSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	PPS	Primary prescaler bits The primary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.

20.10.5 Timer Functional Description

The clock input to the timer prescaler can be selected from the following sources:

- The system clock (ipg_clock)
- The system ‘slow go’ clock (ipg_clock internally divided by 16)

The general system clock is generated in the clock synthesizer and defaults to the system frequency. However, the general system clock has the option to be divided before it leaves the clock synthesizer. This mode, called ‘slow go’ is used to save power. Whatever the resulting frequency of the general system clock, the user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer. Alternatively, the user may prefer the TIN*n* pin to be the clock source. TIN*n* is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding GTMDR*n*[ICLK] bits. The prescalers (GTMDR*n*[SPS] and GTPSR*n*[PPS]) can be programmed to divide the clock input by values from 1 to 65,537 and the output of the prescaler is used as an input to the 16-bit counters. The best resolution of the timer is one clock cycle (5 ns at 200-MHz QUICC Engine reference clock). The maximum period (when the reference value is all ones) for one 16-bit timer is ~83 ms at 200 MHz.

20.10.5.1 Reference Modes

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding GTMDR selects each mode.

- Free run reference mode (GTMDR*n*[FRR]=0):

  The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode (GTMDR*n*[FRR]=1):

  The corresponding timer count is reset immediately after the reference value is reached.

The output can also be internally connected to the input of another timer, resulting in a 32-bit or a 64-bit timer.

20.10.5.2 Capture Modes

The capture modes are disabled in the QUICC Engine Timers.

20.10.5.3 Cascaded Modes

GTCFR*n*[PCAS] and GTCFR2[SCAS] are used to put the timers into different cascaded modes:

- Non-cascaded mode
- Pair-cascaded mode
- Super-cascaded mode

Programming for non-cascaded mode is detailed in Table 20-24.

Table 20-24. Non-Cascaded Mode Programming

GTCFR*n*[PCAS]	GTCFR2[SCAS]	Notes
0	0	Each timer (timer 1, timer 2, timer 3, and timer 4) functions as an independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR and GTCNR for each one (Figure 20-22). When working in the non-cascaded mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with appropriate 16-bit bus cycles.

In non-cascaded mode, each timer functions independently as shown in Figure 20-22.

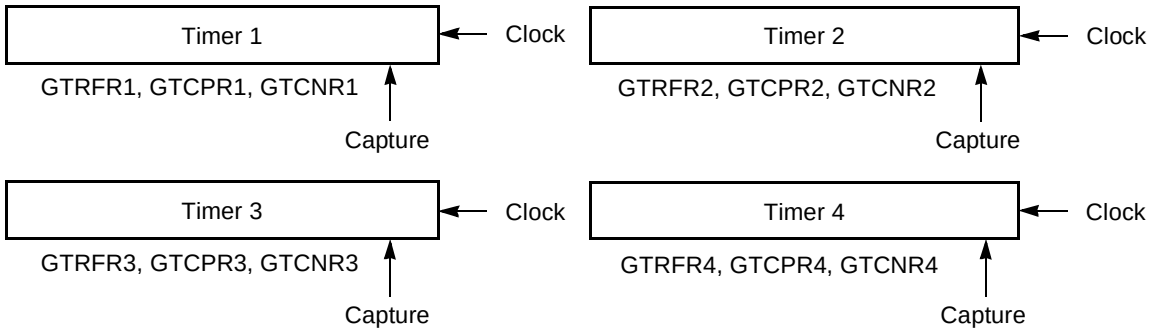


Figure 20-22. Timers Non-Cascaded Mode Block Diagram

Programming for pair-cascaded mode is detailed in Table 20-25.

Table 20-25. Pair-Cascaded Mode Programming

GTCFR1[PCAS]	GTCFR2[PCAS]	GTCF2[SCAS]	Notes
1	0	0	Timer 1 is internally cascaded to Timer 2 to form a 32-bit counter. Timer 3 and Timer 4 are two independent 16-bit timers.
0	1	0	Timer 3 is internally cascaded to Timer 4 to form a 32-bit counter. Timer 1 and Timer 2 are two independent 16-bit timers
1	1	0	Timer 1 is internally cascaded to Timer 2 to form a 32-bit counter. Timer 3 is internally cascaded to Timer 4 to form a second 32-bit counter.

The two 16-bit timers function as a 32-bit timer with a 32-bit GTRFR, GTCPR, and GTCNR. GTMDR1/GTMDR3 is ignored, and the modes and functions are defined using GTMDR2/GTMDR4 and GTCFR1/GTCFR2. The capture is controlled from TIN2/TIN4, and the interrupts are generated from GTEVR2/GTEVR4. The cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.

In the pair-cascade mode, two 16-bit timers can be internally cascaded to form a 32-bit counter as shown in Figure 20-23.

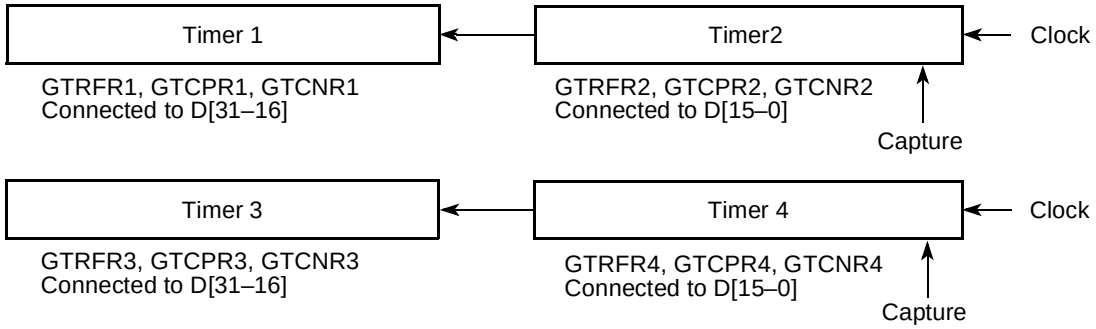


Figure 20-23. Timer Pair-Cascaded Mode Block Diagram

Programming for super-cascaded mode is detailed in Table 20-26.

Table 20-26. Super-Cascaded Mode Programming

GTCFR2[SCAS]	Notes
1	All four 16-bit timers function as a 64-bit timer with a cascaded 32-bit GTRFR, GTCPR, and GTCNR. Registers GTMDR1–3 and GTCFR1 are ignored, and the modes and functions are defined using GTMDR4 and GTCFR2 only. The capture is controlled from TIN4, and the interrupts are generated from GTEVR4. The cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

In super-cascade mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter, as shown in Figure 20-24.

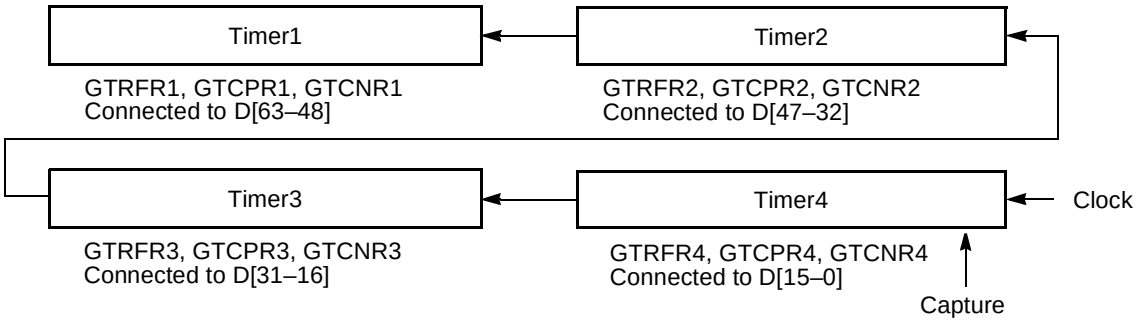


Figure 20-24. Timers Super-Cascaded Mode Block Diagram

20.10.6 Initialization/Application Information

The following initialization sequence is recommended for the GTM registers:

1. Write to GTCFR*n* to reset, to stop or to configure the appropriate timer's operation: cascaded timers configuration, gate mode configuration.
2. Write to GTPSR*n*[PPS] fields to program the appropriate timer clock primary prescaler.
3. Write to GTMDR*n* to choose an input clock, to program the secondary prescaler and to set a desirable appropriate timer's operational mode.

4. Erratic behavior may occur if GTGCR and GTPSR are not initialized before the GTMDR. Only GTGCR[RST] can be modified at any time
5. Clear GTEVR[REF] and GTEVR[CAP] by writing 1's to clear the previous events.
6. Write to GTRFR and to GTCNR*n* according to appropriate timer GTMDR*n* programming.

   A write cycle to a GTCNR*n*[CNV] fields sets the register to the written value, causing its corresponding primary and secondary prescalers, (GTPSR*n*[PPS] and GTMDR*n*[SPS]), to be reset.
7. Write to GTGCR*n*[STP] and to GTGCR*n*[RST] in order to initialize the appropriate timer's operation.

Chapter 21
Serial Peripheral Interface (SPI)

21.1 Introduction

The serial peripheral interface (SPI) allows the exchange of data with other devices containing an SPI. The SPI also communicates with the Ethernet PHY for configuration 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.

The SPI receiver and transmitter are double-buffered, as shown in Figure 21-1, giving an effective FIFO size (latency) of 2 characters. When the SPI is disabled in the SPI mode register (SPMODE[EN] = 0), it consumes little power.

There are two different SPIs. One operates as a normal SPI, and the other is dedicated for the use of MIIMCOM (See Section 21.1.3.4, “SPI in MIIMCOM Mode (Ethernet PHY Management Mode).”

The SPI can operate in QUICC Engine mode or in CPU mode. In QUICC Engine mode SPI is compatible to the MPC826x SPI, and is controlled by QUICC Engine RISC. In CPU mode, the SPI is controlled wholly by the CPU without any QUICC Engine RISC intervention.

21.1.1 Features

The following list summarizes the main features of the SPI:

- Four-signal interface (SPIMOSI, SPIMISO, SPICLK and $\overline{\text{SPISEL}}$)
- Full-duplex operation
- Works with 32-bit data characters, or with a range from 4-bit to 16-bit data characters
- Supports back-to-back character transmission and reception
- Supports master or slave SPI mode
- Supports multiple-master environment
- Continuous transfer mode for automatic scanning of a peripheral
- Maximum clock rate is QUICC Engine clk /8 in master mode and QUICC Engine clk /4 in slave mode (not in back to back operation)
- Independent programmable baud rate generator
- Programmable clock phase and polarity

- Local loopback capability for testing
- Open-drain outputs support multimaster configuration
- Programmable clock gap between two characters in master mode
- Communication with Ethernet PHY for configuration and status (MIIMCOM-MII management communication protocol)
- Multi-MIIMCOM environment with up to 32 PHYs
- Controlled by CPU/QUICC Engine RISC according to user configuration.

21.1.2 Block Diagram

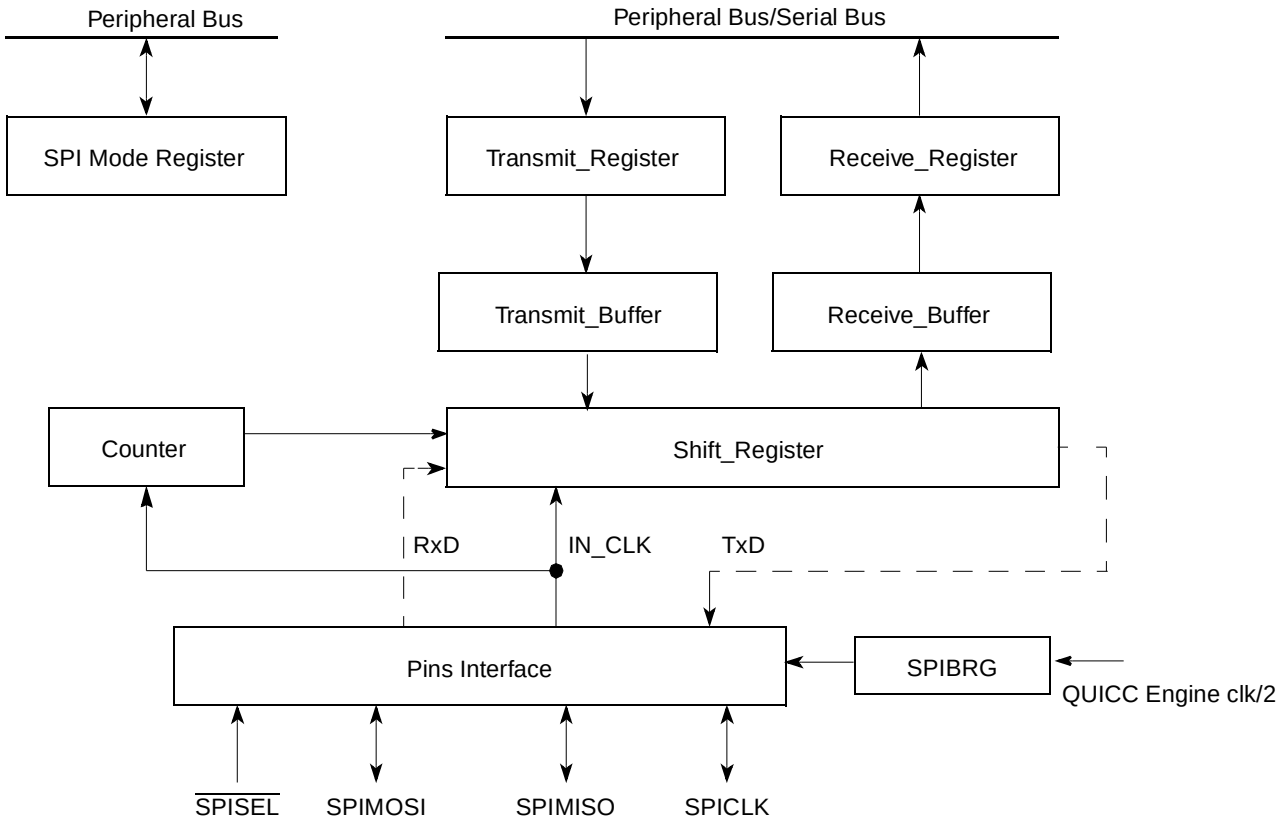


Figure 21-1. SPI Block Diagram

21.1.3 SPI Modes of Operation in QUICC Engine Mode

The SPI can be programmed to work in a single- or multiple-master environment. This section describes the SPI master and slave operation in a single-master configuration and then discusses the multi-master environment and the MIIMCOM mode.

The following sections present a summary of the main modes of operation which the SPI supports.

21.1.3.1 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 device with multiple slaves can use general-purpose parallel I/O signals to selectively enable slaves, as shown in Figure 21-2. To eliminate the multi-master error in a single-master environment, the master's $\overline{\text{SPISEL}}$ input can be forced inactive by selecting $\overline{\text{SPISEL}}$ for general-purpose I/O.

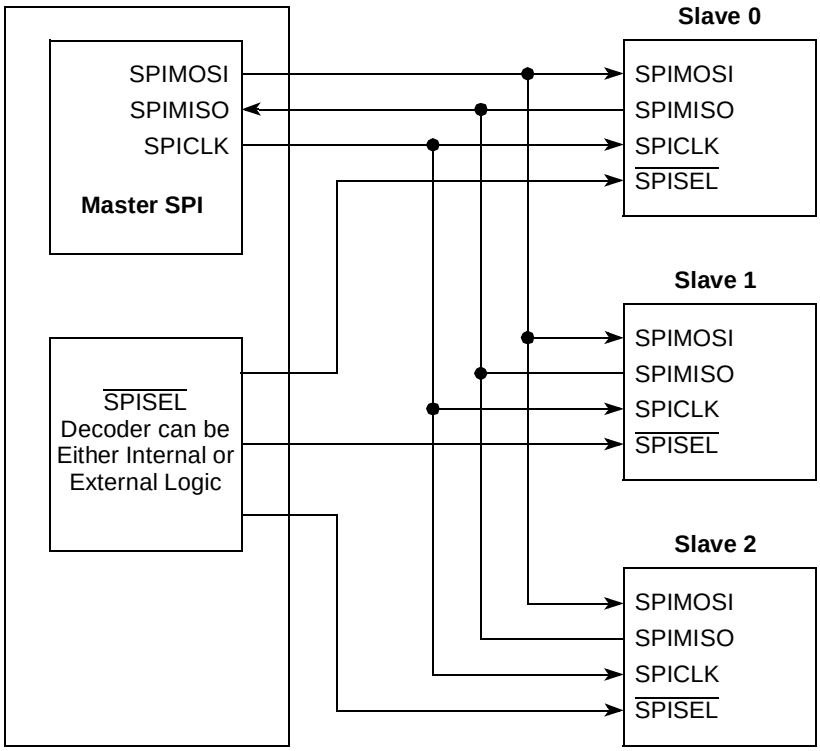


Figure 21-2. Single-Master/Multi-Slave Configuration

To start exchanging data, the QUICC Engine module writes the data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The QUICC Engine module 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 QUICC Engine module then clears TxBD[R] and RxBD[E] and issues a maskable interrupt to the interrupt controller.

When multiple TxBDs are ready, TxBD[L] determines whether the SPI keeps transmitting without SPCOM[STR] being set again. If the current TxBD[L] is cleared, the next TxBD is processed after data from the current buffer is sent. Typically, there is no delay on 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.

21.1.3.2 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 QUICC Engine clk/4.

To prepare for data transfers, the slave's CPU writes data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core processor then sets SPCOM[STR] to activate the SPI. Once $\overline{\text{SPISEL}}$ 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.

NOTE

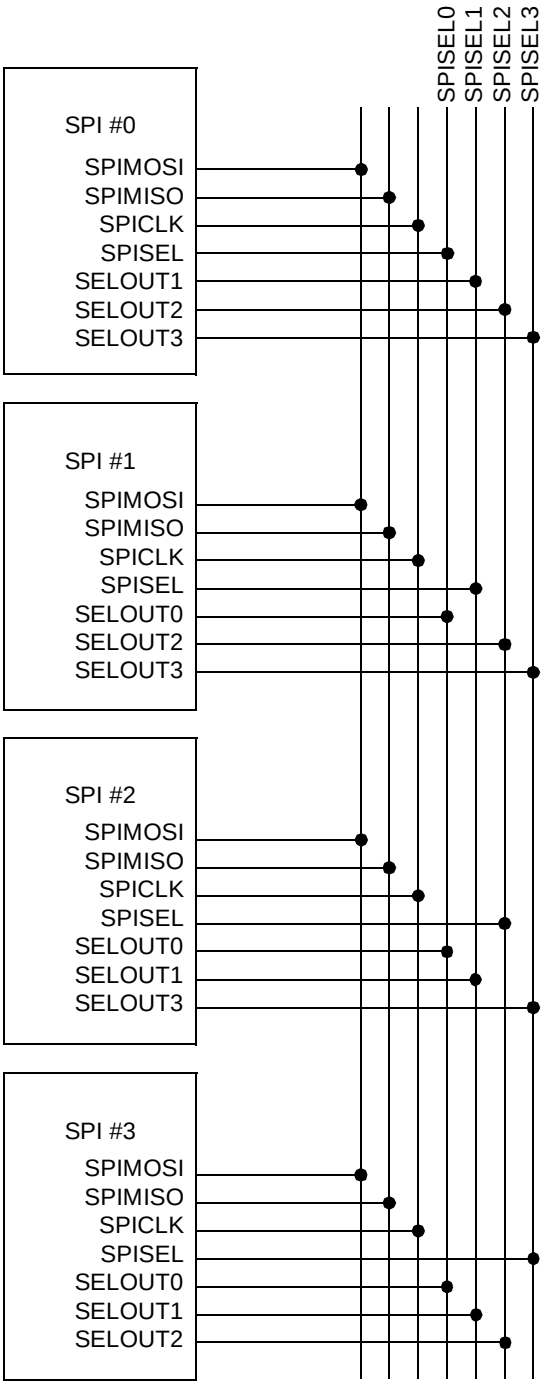
When enabling the SPI or changing parameters in SPI Mode Register (like CP,CI), $\overline{\text{SPISEL}}$ must remain negated for at least 2 QUICC Engine clk/2 clocks afterwards.

Also if $\overline{\text{SPISEL}}$ is negated between transfers, its negation time should be at least 2 QUICC Engine clk/2 clocks.

21.1.3.3 SPI in Multimaster Operation

The SPI can operate in a multi-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 21-3. Only one SPI device at a time can act as a master—all others must be slaves. When an SPI is configured as a master and its $\overline{\text{SPISEL}}$ input is asserted, a multi master error occurs because more than one SPI device is a bus master. The SPI sets SPIE[MME] in the SPI event register and a maskable interrupt is issued to the QUICC Engine module. It also disables SPI operation and the output drivers of SPI signals. The core processor must clear SPMODE[EN] before the SPI is used again. After correcting the problems, clear SPIE[MME] and re-enable the SPI.

The maximum sustained data rate that the SPI supports is QUICC Engine clk/50. However, the SPI can transfer a single character at much higher rates—QUICC Engine clk/8 in master mode and QUICC Engine clk/4 in slave mode. Gaps should be inserted between multiple characters to keep from exceeding the maximum sustained data rate.



Notes:
- All signals are open-drain
- For a multi-master QUICC Engine module with more than two masters, SPISEL and SPIE[MME] will not detect all possible conflicts.
- It is the responsibility of the software to arbitrate for the SPI bus (with token passing, for example).
- SPISELx signals are implemented in the software with general-purpose I/O signals.

Figure 21-3. Multimaster Configuration

21.1.3.4 SPI in MIIMCOM Mode (Ethernet PHY Management Mode)

In MIIMCOM mode, the SPI can receive/transmit data from/to an Ethernet PHY. The Ethernet PHY for or MII has both a MAC interface and a management interface. This section addresses the management interface that contains two lines—MDIO and MDC. These lines are used to manage read and write from/to the PHY internal registers. The SPI should be configured as a master and certain BDs should be specifically configured for this mode.

The PHY communication requires a 2-pin interface (MDIO bi-directional wire for data and MDC input clock) and a special protocol for read and write operations. See Figure 21-4, Figure 21-5, and Figure 21-6 for more information.

21.1.3.4.1 Write Command to PHY Internal Registers

For a write command to PHY internal registers, the QUICC Engine module determines the preamble length desired before the transmit command.

The data pointed by the buffer should contain the following protocol fields:

[START (2’b01)][opcode write (2’b01)][PHY address (5 bits)]
[register address (5 bits)][turn around (2’b10)][data (16 bits)]

RxBD need not be prepared because the SPI does not receive any data when MIIMCOM write command is issued.

21.1.3.4.2 Read Command from PHY Internal Registers

For a read command, the QUICC Engine module determines the preamble length desired before the transmitted command.

The data pointed by the transmit buffer should contain the following protocol fields:

[START (2’b01)][opcode read (2’b10)][PHY address (5 bits)]
[register address (5 bits)][turn around (2’b00)][dont care(16 bits)]

The SPI will transmit the first 14 bits as in the write command, then the MIIMCOM output buffer (see Figure 21-6) is disabled and the PHY drives the desired data towards the SPI.

The last 16 bits in the transmitter are *don’t care* because they are used only for continuing the SPI operation.

The RxBD will contain the first 16 bits which are transmitted by the SPI and should be neglected. The next 16 bits are the received data from the PHY internal registers.

The MDC clock is programmed in the SPMODE register. The other attributes of SPMODE should be configured as: LEN=0x0, REV=1, LOOP=0, CI=0, CP=0, M/S=1 for this mode. The frame structure for the PHY special protocol read and write operations is shown in Figure 21-6.

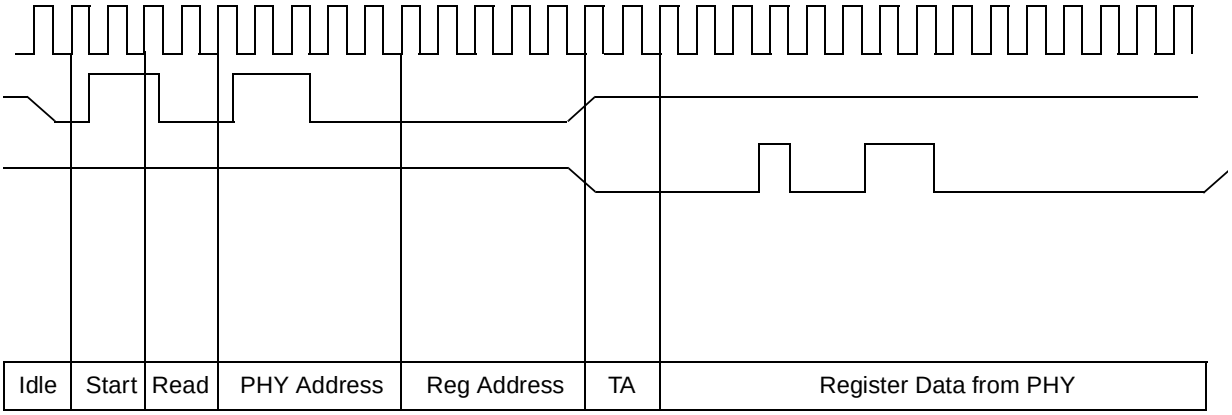


Figure 21-4. Typical Ethernet PHY Management Protocol for Read Operation

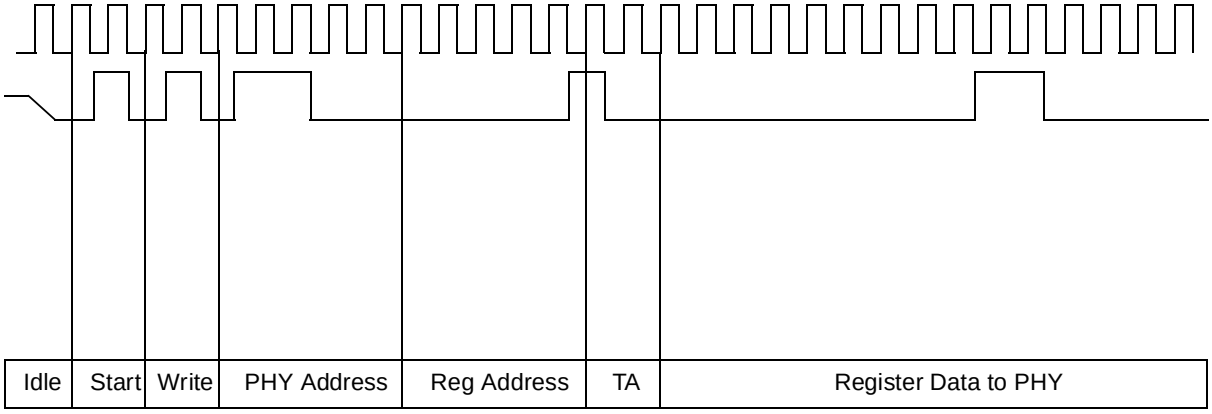


Figure 21-5. Typical Ethernet PHY Management Protocol for Write Operation

0/8/16/32 bit Preamble	Start of Frame 01	Opcode Read/Write 10/01	5-Bit PHY Device Address xxxxx	5-Bit PHY Register Address yyyyy	2-Bit Turn Around Read/Write z0 /10	16-Bit Data Field dddddddddddddddd

Figure 21-6. Serial Management Interface Protocol

Figure 21-7 describes the connectivity to Ethernet PHY.

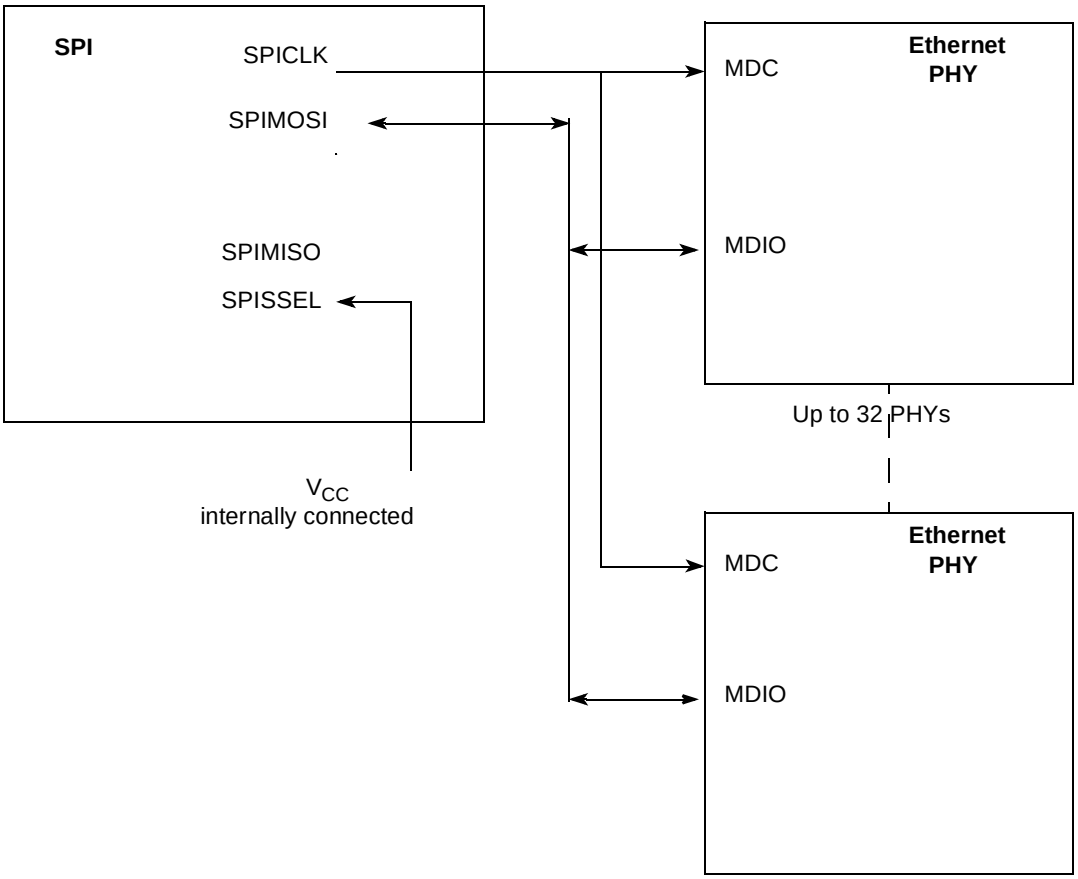


Figure 21-7. MIIMCOM Connectivity to Ethernet PHY

21.2 External Signal Descriptions

The SPI supports a four-wire interface—transmit, receive, clock, and slave select.

21.2.1 Overview

Table 21-1. Signal Properties

Name	Function	Reset	Pull Up
SPIMISO	Master input, slave output	—	Required in open drain mode
SPIMOSI	Master output, slave input, or connected to MDIO in case of MIIMCOM	—	Required in open drain mode
SPICLK	Input/output serial clock connected to the other SPICLK	—	Required in open drain mode
$\overline{\text{SPISEL}}$	SPI slave select	—	Required in open drain mode

21.2.2 Detailed Signal Descriptions

Table 21-2. Detailed Signal Descriptions

Signal	I/O	Description	
SPIMISO	I/O	Master input slave output	
		State Meaning	• Asserted—the data transmitted/received from/to the SPI (depends if master or slave mode) is high • Negated—the data transmitted/received from/to the SPI (depends if master or slave mode) is low
		Timing	• Assertion—according to the SPICLK assertion/negation/in the middle of phase (depends on the SPMODE configuration register). • Negation—according to the SPICLK assertion/negation/in the middle of phase (depends on the SPMODE configuration register)
SPIMOSI	I/O	Master output slave input or connected to MDIO in case of MIIMCOM	
		State Meaning	• Asserted—the data transmitted/received from/to the SPI (depends if master or slave mode) is high • Negated—the data transmitted/received from/to the SPI (depends if master or slave mode) is low
		Timing	• Assertion—according to the SPICLK assertion/negation/in the middle of phase (depends on the SPMODE configuration register). • Negation—according to the SPICLK assertion/negation/in the middle of phase (depends on the SPMODE configuration register).
SPICLK	I/O	Serial clock is input in the slave mode, or output in the master mode.	
		State Meaning	Assertion/Negation according to SPMODE[PM, DIV16] register rate configuration
		Timing	Assertion/Negation—During frame reception/transmission.
$\overline{\text{SPISEL}}$	I	SPI slave select	
		State Meaning	• Asserted—in slave mode declares the slave has been selected for the coming frame; in master mode assertion causes MME multiple-master error • Negated—in slave mode means the specific SPI has not been selected; in master mode needs to be negated for regular operation
		Timing	• Assertion—In slave mode along with the data from the slave. • Negation—In slave mode with the end of the frame (according to SPMODE[LEN]. In master mode before SPCOM[STR] assertion and remains constant.

The SPI can be configured as a slave or as a master in single- or multiple-master environments or in MIIMCOM mode. The master or MIIMCOM SPI generates the transfer clock SPICLK, using the SPI baud rate generator (BRG). The SPI BRG input is QUICC Engine clk /2.

SPICLK is a gated clock, active only during data transfers. Four combinations of SPICLK phase and polarity can be configured by using SPMODE[CI, CP]. SPI signals can also be configured as open-drain to support a multimaster configuration in which a shared SPI signal is driven by the processor or an external SPI device.

The SPI master-in slave-out 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 and is also connected to MDIO in the MIIMCOM configuration. The dual functionality of these signals allows the SPIs in a multimaster environment to communicate with one another using a common hardware configuration.

- When the SPI is a master, 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 a 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 input enable to the SPI slave. In a multi-master environment, $\overline{\text{SPISEL}}$ (always an input) is also used to detect an error when more than one master is operating.
- In case of MIIMCOM, the SPICLK is an output towards the Ethernet PHY.

21.3 Programming the SPI Registers

Table 21-3. SPI Registers Summary in QUICC Engine Mode

Offset from CE_base	Register	Access	Reset Value
SPI1:0x4E0 SPI2: 0x520	SPMODE–SPI mode register	R/W	0x0000_0000
SPI1: 0x4E4 SPI2: 0x524	SPIE–SPI event register	R/W	0x00
SPI1: 0x4E8 SPI2: 0x528	SPIM–SPI mask register	R/W	0x00
SPI1:0x4EDC SPI2: 0x52C	SPCOM–SPI command register	W	0x00

21.3.1 SPI Mode Register (SPMODE)

The SPI mode register (SPMODE), shown in Figure 21-8, controls both the SPI operation mode and clock source.

Offset: SPI1: 0x4E0
SPI2: 0x520

Access: Read/Write

Bit	0	1	2	3	4	5	6	7	8–11	12–15
R W	EM	LOOP	CI	CP	DIV16	REV	M/S	EN	LEN	PM
Reset	All zeros									

Bit	16	17	18	19	20–24	25–31
R W	—	OP	MII	—	CG	—
Reset	All zeros					

Figure 21-8. SPMODE-SPI Mode Register

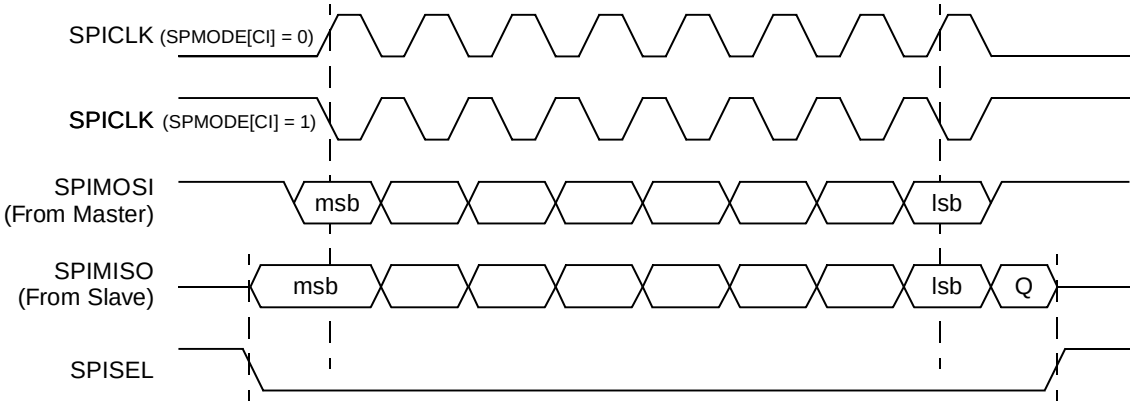
Table 21-4 describes the SPMODE fields.

Table 21-4. SPMODE Field Descriptions

Bits	Name	Description
0	EM	Emergency request to QUICC Engine RISC 0 Emergency requests to QUICC Engine RISC (to prevent underrun/overrun) 1 Normal requests to QUICC Engine RISC
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 21-9 and Figure 21-10. 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 21-9 and Figure 21-10. 0 SPICLK starts toggling at 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 QUICC Engine clk/2 is the input to the SPI BRG 1 QUICC Engine clk/32 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/ functions in MIIMCOM mode
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.
8–11	LEN	Character length in bits per character. Must be between 0011 (4 bits) and 1111 (16 bits) or for 32 bits character the length should be 0000. A value less than 4 (except 0) causes erratic behavior. If the value is not greater than a byte, every **byte** in memory holds (LEN + 1) valid bits. If the value is greater than a byte, every **half-word** holds (LEN + 1) valid bits. For 32 bits every word in memory holds 32 valid bits. See Section 21.3.1.1, “SPI Examples with Different SPMODE[REV,LEN] Values.”
12–15	PM	Prescale modulus select. Specifies the divide ratio of the prescale divider in the SPI clock generator. QUICC Engine clk/2 (or QUICC Engine clk/32 according to DIV16 bit) is divided by 4 × ([PM0–PM3] + 1), a range from 4 to 64. The clock has a 50% duty cycle.
16	—	Reserved, should be cleared
17	OP	Operation mode 0 QUICC Engine mode—the SPI is controlled by the QUICC Engine RISC 1 CPU mode—the SPI is controlled by the CPU
18	MII	MIICOM mode 0 SPI works as a regular SPI 1 SPI works in MIICOM mode
19	—	Reserved, should be cleared

Table 21-4. SPMODE Field Descriptions (continued)

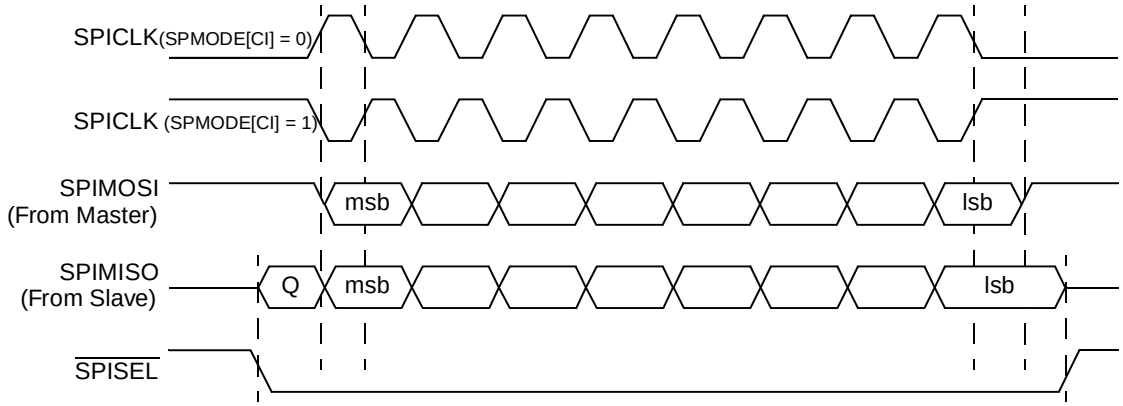
Bits	Name	Description
20–24	CG	Clock Gap For SPI Master mode: insert gaps between two transmitted characters according to CG size. Example: CG = 00101 inserts 5 bits time gap between every two consecutive characters For MIICOM mode: Preamble length - (only these values are allowed) 00000 send 0 IDLE bits before frame transmission 00100 send 8 IDLE bits before frame transmission 01000 send 16 IDLE bits before frame transmission 10000 send 32 IDLE bits before frame transmission
25–31	—	Reserved, should be cleared



Note: Q = Undefined signal.

Figure 21-9. SPI Transfer Format with SPMODE[CP] = 0

Figure 21-10 shows the SPI transfer format in which SPICLK starts toggling at the beginning of the transfer (SPMODE[CP] = 1).



Note: Q = Undefined signal.

Figure 21-10. SPI Transfer Format with SPMODE[CP] = 1

21.3.1.1 SPI Examples with Different SPMODE[REV,LEN] Values

The examples below show how SPMODE[REV,LEN] affect character transmission on the line. For all examples below, assume the memory (Big Endian) contains the following binary image and one character is transmitted:

```
[address%4=0x0_0x1_0x2_0x3] ghijklmn__opqrstuv_wxyzabcd_efGHIJKL
```

Example 21-1.

```
with LEN=4 (data size=5),REV=0, the string transmitted is:
         first_bit nmlkj last_bit
with REV=1,the string transmitted is:
         first_bit jklmn    last_bit
```

Example 21-2.

```
with LEN=7 (data size=8),REV=0, the string transmitted is:
         first_bit nmlkjihg last_bit
with REV=1, the string transmitted is:
         first_bit ghijklmn last_bit
```

Example 21-3.

```
with LEN=0xC (data size=13),REV=0, the string transmitted is:
         first_bit nmlkjihgvutsr last_bit
with REV=1, the string transmitted is:
         first_bit rstuvghijklmn last_bit
```

Example 21-4.

```
with LEN=0xF (data size=16), REV=0, the string transmitted is:
         first_bit nmlkjihgvutsrqpo last_bit
with REV=1, the string transmitted is:
         first_bit opqrstuvghijklmn last_bit
```

Example 21-5.

```
with LEN=0 (data size=32), REV=0, the string transmitted is:
         first_bit nmlkjihgvutsrqpodcbazyxwLKJIHGfe last_bit
with REV=1, the string transmitted is:
         first_bit efGHIJKLwxyzabcdopqrstuvghijklmn last_bit
```

21.3.2 SPI Event/Mask Registers (SPIE/SPIM)

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 a 0 has no effect. Setting a bit in the SPIM enables the corresponding interrupt and clearing a bit masks it. Unmasked SPIE bits must be

cleared before the QUICC Engine module clears internal interrupt requests. Figure 21-11 shows the SPIE register.

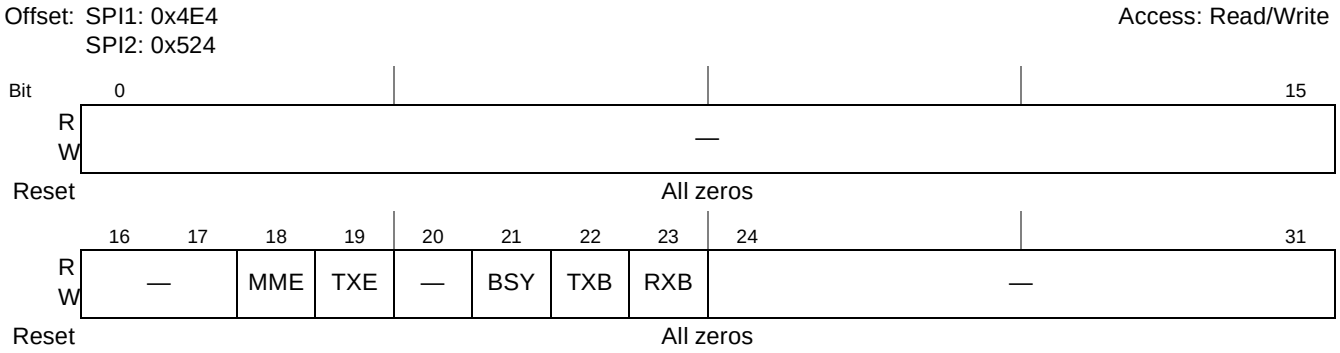


Figure 21-11. SPI Event (SPIE) Register

Table 21-5 describes the SPIE fields.

Table 21-5. SPIE Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared
18	MME	Multimaster error. Set when $\overline{\text{SPISEL}}$ is asserted externally while the SPI is in master mode.
19	TXE	Tx error. Set when an error occurs during transmission.
20	—	Reserved, should be cleared
21	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.
22	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.
23	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the BD is closed.
24–31	—	Reserved, should be cleared

Figure 21-12 shows the SPIM register.

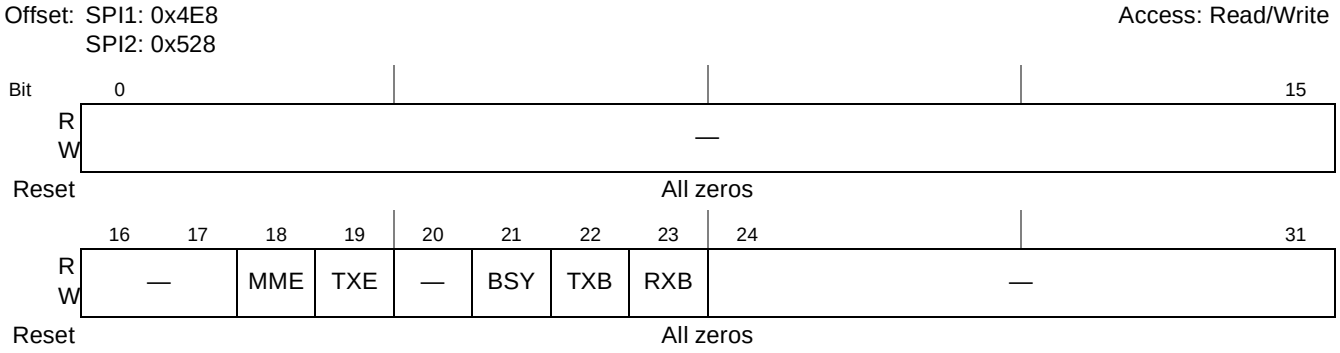


Figure 21-12. SPI Mask (SPIM) Register

Table 21-6 describes the SPIM fields.

Table 21-6. SPIM Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared
18	MME	Multimaster error. Masks multimaster error interrupt. 0 Interrupt masked 1 Interrupt enabled
19	TXE	Tx error. Masks transmit error interrupt. 0 Interrupt masked 1 Interrupt enabled
20	—	Reserved, should be cleared
21	BSY	Busy. Masks SPI busy interrupt. 0 Interrupt masked 1 Interrupt enabled
22	TXB	Tx buffer. Masks transmit buffer interrupt. 0 Interrupt masked 1 Interrupt enabled
23	RXB	Rx buffer. Masks receive buffer interrupt. 0 Interrupt masked 1 Interrupt enabled
24–31	—	Reserved, should be cleared

21.3.3 SPI Command Register (SPCOM)

SPCOM, shown in Figure 21-7, is used to start SPI operation.

Offset: SPI1: 0x4EC
SPI2: 0x52C

Access: Write only

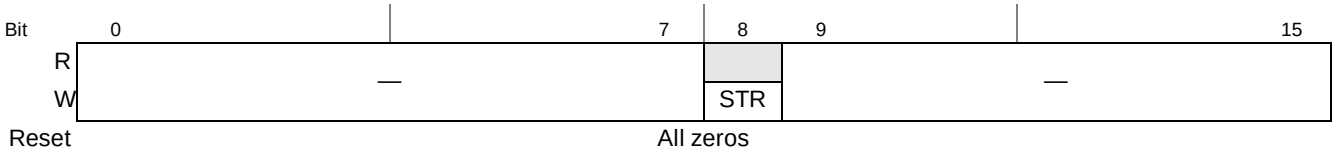


Table 21-7. SPI Command Register (SPCOM)

Table 21-8 describes the SPCOM fields.

Table 21-8. SPCOM Field Descriptions

Bits	Name	Description
1–7	—	Reserved
8	**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 QUICC Engine/2 clock cycle. 0 SPI Idle 1 Start SPI transaction
9–15	—	Reserved. Should be cleared

21.4 SPI Parameter RAM

Some values must be user-initialized before the SPI is enabled; the QUICC Engine module 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 21-9 shows the memory map of the SPI parameter RAM.

Table 21-9. SPI Parameter RAM Memory Map

Offset[1]	Name[2]	Width	Description
0x00	**RBASE**	Hword	Rx/Tx BD table base address. Indicate where the BD tables begin in the multiuser 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	**RX BUS MODE**	Byte	Rx/Tx Bus Mode registers. They contain the transaction specification associated with DMA channel accesses to external memory. See Section 21.4.1, "Receive/Transmit Bus Mode Registers."
0x05	**TX BUS MODE**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. The SPI has one MRBLR entry to define the maximum number of bytes the QUICC Engine module writes to an Rx buffer before moving to the next buffer. The QUICC Engine module can write fewer bytes than MRBLR if an error or end-of-frame occurs, but never exceeds the MRBLR value. User-supplied buffers should not be 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 QUICC Engine module 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. It should be divisible by 4 if the character length of the data is 32 bits.
0x08–0x0F	—	Word	Reserved for QUICC Engine module use.

Table 21-9. SPI Parameter RAM Memory Map (continued)

Offset[1]	Name[2]	Width	Description
0x10	RBPTR	Hword	RxBD pointer. Points to the current Rx BD 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 QUICC Engine module 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.
0x12–0x1F	—	Hword	Reserved for QUICC Engine module use.
0x20	TBPTR	Hword	TxBD pointer. Points to the current Tx BD during frame transmission or the next BD to be processed when idle. After reset or when the end of the Tx BD table is reached, the QUICC Engine module 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.
0x22–0x4B	—	Word	Reserved for QUICC Engine module use.

1 From SPI page base address in Multiuser RAM.
2 **The user must initialize only items in bold.**

21.4.1 Receive/Transmit Bus Mode Registers

Figure 21-13 shows the fields in the receive/transmit bus mode registers. Bolded bits must be configured by the user.

Offset: Rx: 0x04
Tx: 0x05

Access: Read/Write

Bits	0	1	2	3	4	5	6	7
R W	—		GBL	BO		CETM	DTB	—

Figure 21-13. Rx/Tx Bus Mode Registers

Table 21-10 describes the Rx/Tx bus mode register fields.

Table 21-10. Rx/Tx Bus Mode Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be 0
2	GBL	Global. Indicates whether the memory operation should be snooped. 0 Snooping on the system bus is disabled. 1 Snooping on the system bus is enabled. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
3–4	BO	Byte ordering. Used to select the byte ordering of the buffer. 00, 01, 11 Reserved 10 Big-endian byte ordering. As data is sent onto the serial line from the data buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same buffer word.

Table 21-10. Rx/Tx Bus Mode Register Field Descriptions (continued)

Bits	Name	Description
5	CETM	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine module, for debug purposes.
6	DTB	Indicates on what bus the data is located. 0 On the system bus 1 On the QUICC Engine secondary bus.

21.5 SPI Buffer Descriptor (BD) Table

As shown in Figure 21-14, BDs are organized into separate RxBD and TxBD tables in multi-user RAM. The tables have the same basic configuration as for the UCCs and form circular queues that determine the order buffers are transferred. The QUICC Engine block uses BDs to confirm reception and transmission or to indicate error conditions so that the core processor knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the multiuser RAM.

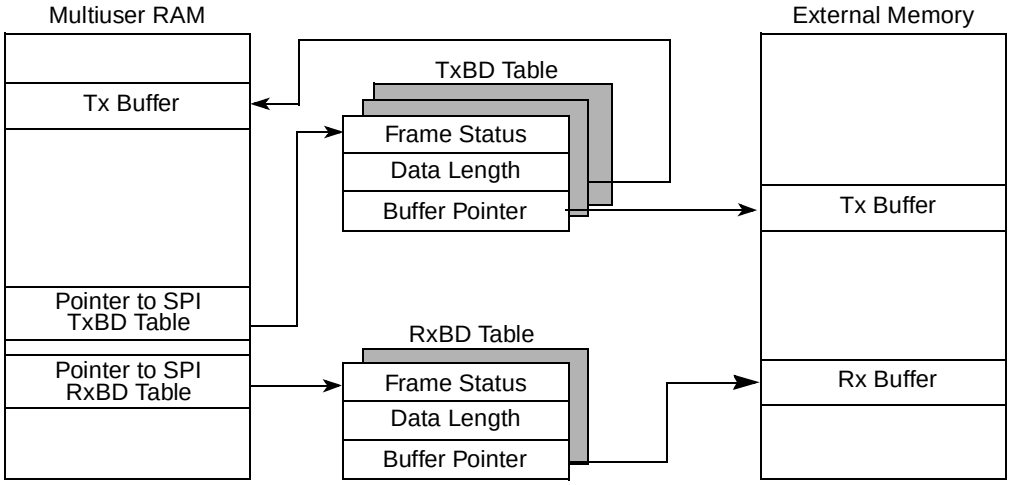


Figure 21-14. SPI Memory Structure

21.5.1 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 21-16 and Figure 21-17, has the following structure:

The SPI uses the basic BD structure of the QUICC Engine subsystem as shown in Figure 21-15.

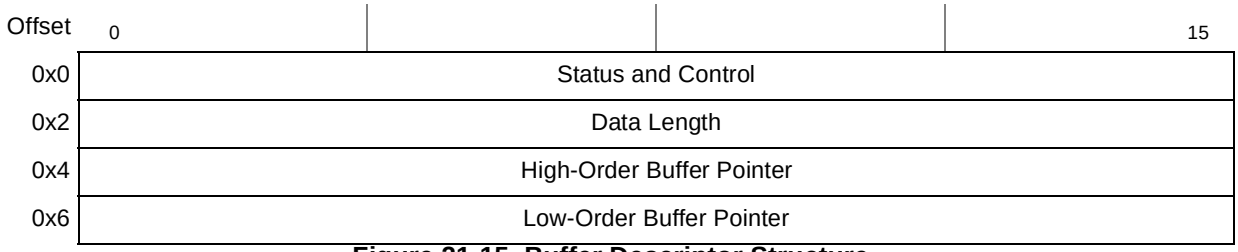


Figure 21-15. Buffer Descriptor Structure

Table 21-11 describes the buffer descriptor fields.

Table 21-11. Buffer Descriptor Field Descriptions

Offset	Name	Description
0x00	Status and Control	Status and Control. The QUICC Engine module updates the status bits after the buffer is sent or received
0x02	Data Length	Data Length that is sent or received. For an RxBD, this is the number of octets the QUICC Engine module writes into this RxBD's buffer once the BD closes. The QUICC Engine module 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 QUICC Engine module 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 3 characters of 8-bit data, the data length field should be initialized to 3. However, to send 3 characters of 9-bit data, the data length field should be initialized to 6, because the three 9-bit data fields occupy 3 half-words in memory. If the character length is 32 bits, the data length should be divisible by 4. The QUICC Engine module never modifies this field.
0x04	High-Order Buffer Pointer	High-Order Buffer Pointer 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.
0x06	Low-Order Buffer Pointer	Low-Order Buffer Pointer 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.

21.5.1.1 SPI Receive BD (RxBD)

The QUICC Engine module 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 QUICC Engine module also closes the buffer when the SPI is configured as a slave and $\overline{\text{SPISEL}}$ is

negated, indicating that reception stopped. The core processor should write RxBD bits before the SPI is enabled. The format of an RxBD is shown in Figure 21-16.

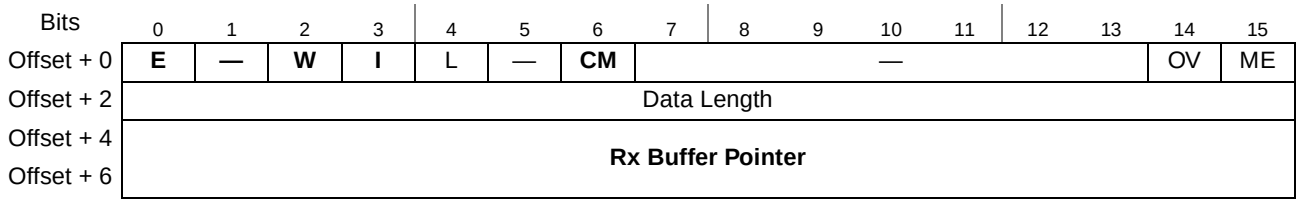


Figure 21-16. SPI RxBD

Table 21-12 describes the RxBD status and control fields.

Table 21-12. SPI 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 processor can examine or write to any fields of this RxBD, but the QUICC Engine module does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The QUICC Engine module owns this RxBD and its buffer. Once E is set, the core processor 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 QUICC Engine module 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 multiuser 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 processor to process the buffer. SPIE[RXB] causes an interrupt if not masked.
4	L	Last. Updated by SPI in slave mode when the buffer is closed because $\overline{\text{SPISEL}}$ was negated or overrun occurred. Updated by SPI in master mode when the buffer is closed because it contains the last character of the message or MME or overrun occurred. 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.
5	—	Reserved, should be cleared
6	CM	Continuous mode. Master mode only; in slave mode, CM should be cleared. 0 Normal operation 1 The QUICC Engine module does not clear RxBD[E] after this BD is closed; the buffer is overwritten when the QUICC Engine module 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 processor overhead. However, the E-bit is cleared if an error (overrun or MME) occurs during reception, regardless of the CM bit.
7–13	—	Reserved, should be cleared

Table 21-12. SPI RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
14	OV	Overrun. Set when a receiver overrun occurs during reception. Old received data is kept while the new data received is discarded. 0 Normal Operation 1 Overrun Occurred
15	ME	Multimaster 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 channel. 0 Normal Operation 1 Multimaster error occurred

21.5.1.2 SPI Transmit BD (TxBD)

Data to be sent with the SPI is sent to the QUICC Engine module by arranging it in buffers referenced by TxBDs in the TxBD table. TxBD fields should be prepared before data is sent. The format of a TxBD is shown in Figure 21-17.

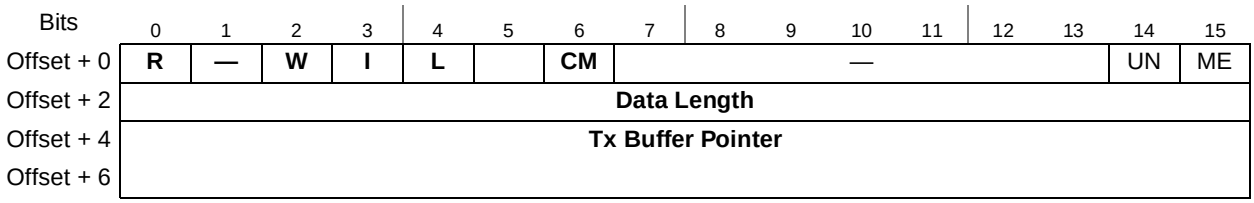


Figure 21-17. SPI TxBD

Table 21-13 describes the TxBD status and control fields.

Table 21-13. SPI TxBD Status and Control Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer is not ready to be sent. This BD or its buffer can be modified. The QUICC Engine module 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 QUICC Engine module 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 multi-user RAM.
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. For MIICOM mode Last should be set only when the last buffer word is MII read command.

Table 21-13. SPI TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
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 QUICC Engine module does not clear TxBD[R] after this BD is closed, allowing the buffer to be resent automatically when the QUICC Engine module next accesses this BD. However, the R-bit is cleared if an error (MME or underrun) occurs during transmission or if L -bit is set, regardless of the CM bit.
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. 0 Normal operation 1 Underrun occurred
15	ME	Multimaster 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. 0 Normal operation (no Multimaster error) 1 Multi-master error occurred

21.6 SPI Commands

Table 21-14 lists transmit/receive commands sent to the QUICC Engine command register (CECR).

Table 21-14. 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.
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.

21.7 SPI Programming Examples

21.7.1 SPI Master Programming Example

The following sequence initializes the SPI to run at a high speed in master mode:

1. Configure I/O ports 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. Write RBASE and TBASE in the SPI parameter RAM to point to the RxBD and TxBD tables in the multi-user RAM, write RBASE with 0x0000 and TBASE with 0x0008.
4. Write Rx/Tx Bus Mode registers.
5. Write MRBLR with the maximum number of bytes per Rx buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.

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 to CECR.
9. Write x0FF to SPIE to clear any previous events.
10. Write 0x37 to SPIM to enable all possible SPI interrupts.
11. Write 0x0370 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, 8-bit characters, and the fastest speed possible.
12. 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.

21.7.2 SPI Slave Programming Example

The following is an initialization sequence example 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.

1. Configure I/O ports to enable SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$.
2. Assuming one RxBD at the beginning of the multiuser RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008 in the SPI parameter RAM.
3. Write Rx/Tx Bus Mode registers.
4. Set MRBLR = 0x0010 for 16 bytes, the maximum number of bytes per buffer.
5. 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].
6. 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].
7. Execute the INIT RX AND TX PARAMETERS command by writing to CECR.
8. Write 0xFF to SPIE to clear any previous events.
9. Write 0x37 to SPIM to enable all SPI interrupts.
10. Set SPMODE to 0x0170 to enable normal operation (not loopback), slave mode, SPI enabled, and 8-bit characters. Baud-rate generator speed is ignored in slave mode.
11. Set SPCOM[STR] to enable the SPI to be ready once the master begins to transfer.

Note that 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.

21.7.3 SPI MIIMCOM Programming Example

The following sequence initializes the SPI to execute a read command in MIIMCOM mode:

1. Configure I/O ports to enable SPIMOSI, SPICLK.
2. Write RBASE and TBASE in the SPI parameter RAM to point to the RxBD and TxBD tables in the multi-user RAM, write RBASE with 0x0000 and TBASE with 0x0008.
3. Write Rx/Tx Bus Mode registers.
4. For MIIMCOM, the MRBLR must be 4 bytes, because this is the Ethernet PHY's frame size for read and write operations, so MRBLR = 0x0004.
5. 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]. This buffer is allocated for a read command, therefore, the first 16 bits received should be neglected. The last 16 bits are the received data from the PHY's register.
6. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2008 in main memory and contains four 8-bit characters. Write 0xB800 (read command) to TxBD[Status and Control], 0x0004 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer] which contains the read command and 2 bytes that will not be transmitted (just for the SPI's operation during the data reception from the PHY).
7. Place the Tx data for the read command: 0x0000_2000 and 0x0000_2001 data is irrelevant.0x0000_2003={8'b0110,PHYAD[0:3]}, 0x0000_2002={PHYAD[4],REGAD[0:4],00} where PHYAD[0] and REGAD[0] are the MSBs of the PHY address and register address respectively.
8. Execute the INIT RX AND TX PARAMETERS command by writing to CECR.
9. Write 0x0FF to SPIE to clear any previous event.
10. Write 0x37 to SPIM to enable all possible SPI interrupts.
11. Write to SPMODE[PM,DIV16] the desired clock rate which should be less than the frequency limit of the PHY.
12. Set the following bits to the indicated values:
    - 12.a. SPMODE[REV]=1
    - 12.b. SPMODE[M/S]=1
    - 12.c. SPMODE[CP]=0
    - 12.d. SPMODE[CI]=0
    - 12.e. SPMODE[LOOP]=0
    - 12.f. SPMODE[LEN]=0x0
13. Set SPCOM[STR] to start the transfer.

After 4 bytes are sent, the TxBD is closed. Additionally, the Rx buffer is closed after 4 bytes are received because TxBD[L] is set.

21.8 Handling Interrupts in the SPI

The following sequence should be followed to handle interrupts in the SPI:

1. When 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. Clear the interrupt by writing a one (1) to SPI's bit in the pending register in the interrupt controller.
4. Execute an **rfi** instruction.

21.9 SPI in CPU Mode

NOTE

SPI in CPU Mode applies to MPC8360E and MPC8568E only.

Table 21-15. SPI Registers Summary in CPU Mode

Offset from CE_base	Register	Access	Reset Value
SPI1: 0x4E0 SPI2: 0x520	SPMODE–SPI mode register	R/W	0x0000_0000
SPI1: 0x4E6 SPI2: 0x526	SPIE–SPI event register	R/W	0x00
SPI1: 0x4EA SPI2: 0x52A	SPIM–SPI mask register	R/W	0x00
SPI1: 0x4ED SPI2: 0x52D	SPCOM–SPI command register	W	0x00
SPI1: 0x4F0 SPI2: 0x530	SPITD–SPI transmit register	W	0x0000_0000
SPI1: 0x4F4 SPI2: 0x534	SPIRD–SPI receive register	R	0xFFFF_FFFF

21.9.1 SPI Transmission and Reception Process

As the SPI is a character-oriented communication unit, the CPU is responsible for packing and unpacking the receive/transmit frames. A frame consists of all of the characters transmitted or received during a completed SPI transmission session, from the first character written to the SPI transmit data register (SPITD) to the last character transmitted following the setting of the LST bit in the SPCOM register.

The CPU receives data by reading the SPI receive data register (SPIRD) when the NE ("not empty") bit in the SPI event register (SPIE) is set.

The CPU transmits data by writing it into the SPITD. When the next character to be transmitted is going to be the final one in the current frame, the CPU sets the "last" (LST) bit in the SPI command register (SPCOM), and then writes the final character to SPITD.

The SPI sets the NF (“not full”) bit in SPIE whenever its transmit FIFO is not full. It clears it when the data indicated by LST is written to SPITD, and re-sets it after sending the last data.

The SPI CPU handshake protocol can be implemented by using a polling or interrupt mechanism. When using a polling mechanism, the CPU reads the SPIE in a predefined frequency and acts according to the value of the SPIE bits. The polling frequency depends on the SPI serial channel frequency. When using the interrupt mechanism, setting either the NF (not full) or NE (not empty) bits of the SPIE causes an interrupt to the CPU. The CPU then reads the SPIE and acts appropriately. There are three basic modes of operation for transmitting and receiving: master, slave, and multimaster.

21.9.2 SPI Mode Register (SPMODE) in CPU Mode

SPMODE, shown in Figure 21-18, controls both the SPI operation mode and clock source.

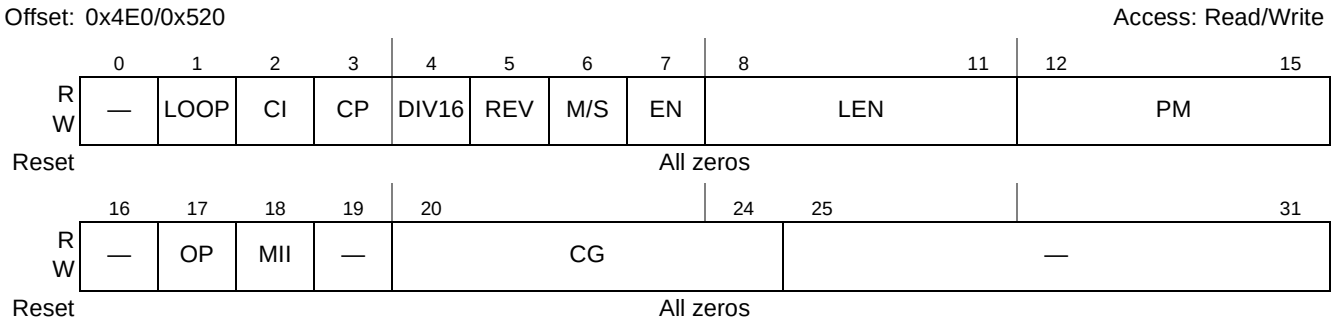


Figure 21-18. SPMODE-SPI Mode Register in CPU Mode

Table 21-16 describes the SPMODE fields.

Table 21-16. SPMODE Field Descriptions

Bits	Name	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 21-9 and Figure 21-10. 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 21-9 and Figure 21-10. 0 SPICLK starts toggling at 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. This bit should be 0 in slave mode. 0 QUICC Engine clk/2 is the input to the SPI BRG 1 QUICC Engine clk/32 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.

Table 21-16. SPMODE Field Descriptions (continued)

Bits	Name	Description
6	M/S	Master/slave. Selects master or slave mode. 0 The SPI is a slave 1 The SPI is a master/ functions in MIIMCOM mode
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.
8–11	LEN	Character length in bits per character. Must be between 0011 (4 bits) and 1111 (16 bits) or for 32 bits character the length should be 0000. A value less than 4 (except 0) causes erratic behavior. Note that the transmit and receive registers each can hold only one character regardless of the character length. SPITD - REV = 0 - lsb should be in bit 31. SPITD - REV = 1 - msb should be in bit 0. SPIRD - REV = 0 - for LEN = 0000 msb is in bit 0. for other LEN values - msb is in bit 16. SPIRD - REV = 1 - for LEN = 0000 lsb is in bit 31. for other LEN values - lsb is in bit 15.
12–15	PM	Prescale modulus select. Specifies the divide ratio of the prescale divider in the SPI clock generator. QUICC Engine clk/2 (or QUICC Engine clk/32 according to DIV16 bit) is divided by 4 × ([PM0–PM3] + 1), a range from 4 to 64. The clock has a 50% duty cycle.
16	—	Reserved, should be cleared
17	OP	Operation mode 0 QUICC Engine mode: the SPI is controlled by the QUICC Engine RISC **1 CPU mode: the SPI is controlled by the CPU**
18	MII	MIICOM mode 0 SPI works as a regular SPI 1 SPI works in MIICOM mode
19	—	Reserved, should be cleared
20–24	CG	Clock Gap For SPI Master mode: insert gaps between two transmitted characters according to CG size. Example: CG = 00101 inserts 5 bits time gap between every two consecutive characters
25–31	—	Reserved, should be cleared

21.9.3 SPI Event Register (SPIE) in CPU MODE

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 CPU clears internal interrupt requests.

Bits NE and NF are status bits. They are not cleared as a result of writing to SPIE. Figure 21-19 shows SPI event register.

Offset: 0x4E6/0x526 Access: Mixed

	16	17	18	19	20	21	22	23
R	—	LT	—	OV	UN	MME	NE	NF
W								
Reset	All zeros							

Figure 21-19. SPI Event Register (SPIE) in CPU Mode

Table 21-17 describes the SPIE fields in CPU mode.

Table 21-17. SPIE Field Descriptions

Bits	Name	Description
16	—	Reserved, should be cleared.
17	LT	Last character was transmitted. The last character was transmitted and new data can be written to the SPITD register for further transmission.
18	—	Reserved, should be cleared.
19	OV	Overrun This bit indicates that an overrun has occurred during reception. In case of overrun the SPIRD contains old received data (it is not overriden by the new data). Overrun is reported for the missing characters. SPI continues transmission/reception in overrun.
20	UN	Slave underrun. This bit indicates that the SPI transmitter does not have data to transmit on time. Occurs only in slave mode (SPMODE[M/S]) = 0. In case of underrun the transmit FIFO is flushed.
21	MME	Multiple-master error. Set when SPISEL is asserted externally while the SPI is in master mode. Note that the MME error can occur in loopback mode.
22	NE	Not empty. Indicates that the SPIRD register contains a received character. 0 The SPIRD is empty 1 The SPIRD has a received character. The CPU can read the content of SPIRD.
23	NF	Not full. When set Indicates that a new character can be written to the transmitter by the CPU. 0 The transmitter FIFO is full. 1 The transmitter FIFO is not full. The CPU is free to write to SPITD.

21.9.4 SPI Mask Register (SPIM) in CPU Mode

SPIM enables/masks interrupts for events recognized by the SPI. When an event is recognized, the SPI sets the corresponding SPIE bit. 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 CPU clears internal interrupt requests. The NE and NF bits in SPIE are status bits. They are not cleared as a result of writing to SPIE. Figure 21-20 shows SPI Mask register.

Offset: 0x4EA/0x52A Access: Read/Write

	16	17	18	19	20	21	22	23
R W	—	LT	—	OV	UN	MME	NE	NF
Reset	All zeros							

Figure 21-20. SPI Mask Register (SPIM) in CPU Mode

Table 21-18 describes the SPIM fields.

Table 21-18. SPIM Field Descriptions

Bits	Name	Description
16	—	Reserved, should be cleared.
17	LT	Last character transmitted 0 LT event will not cause SPI Interrupt 1 LT event will cause SPI Interrupt
18	—	Reserved, should be cleared.
19	OV	Overrun interrupt mask 0 Overrun event will not cause SPI Interrupt 1 Overrun event will cause SPI Interrupt
20	UN	Slave Underrun interrupt mask 0 Slave Underrun event will not cause SPI Interrupt 1 Slave Underrun event will cause SPI Interrupt
21	MME	Multimaster error interrupt mask 0 Multimaster error event will not cause SPI Interrupt 1 Multimaster error event will cause SPI Interrupt
22	NE	Not Empty interrupt mask 0 Not Empty event will not cause SPI Interrupt 1 Not Empty event will cause SPI Interrupt
23	NF	Not Full interrupt mask 0 Not Full event will not cause SPI Interrupt 1 Not Full event will cause SPI Interrupt

21.9.5 SPI Command Register (SPCOM) in CPU Mode

SPCOM, shown in Figure 21-21, indicates whether the next character to be written to SPITD is the last character of the frame.

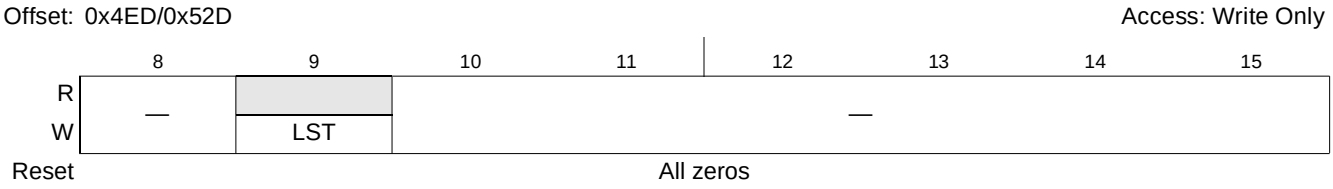


Figure 21-21. SPI Command Register (SPCOM) in CPU Mode

Table 21-19 describes the SPCOM fields.

Table 21-19. SPCOM Field Descriptions

Bits	Name	Description
8	—	Reserved. Should be cleared.
9	**LST**	This bit indicates if the next character written to SPITD is the last one of the frame. 0 The next character written to SPITD is not the last character of the frame 1 The next character written to SPITD is the last character of the frame
10–15	—	Reserved. Should be cleared.

21.9.6 SPI Transmit Data Register (SPITD)

SPITD holds the character to be transmitted. The number of bits in each character is specified by SPMODE[LEN]. In slave mode the CPU should write the first character to SPITD before $\overline{\text{SPISEL}}$ is asserted. Each time the NF bit in the SPIE is set, the CPU can write another character of data to the SPITD register, if there is no error indication in the SPIE. At the end of the frame the CPU should set the SPCOM[LST] bit and write the last character of data. SPITD must be written as 32 bits for any character length size. Figure 21-22 shows the SPI transmit data register.

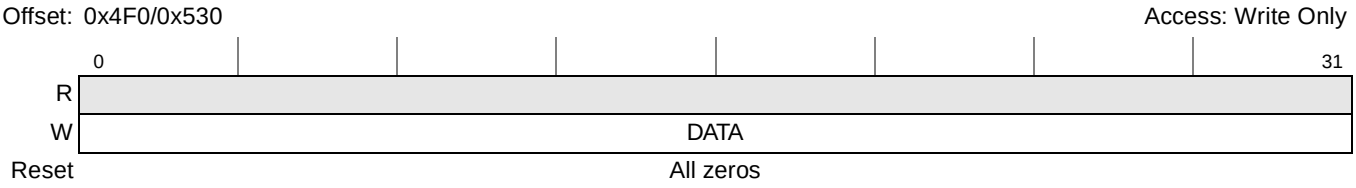


Figure 21-22. SPI Transmit Data Register (SPITD)

21.9.7 SPI Receive Data Register (SPIRD)

SPIRD holds a received character. Each time the SPIE[NE] bit is set, the CPU can read the SPIRD.

Offset: 0x4F4/0x534 Access: Read Only

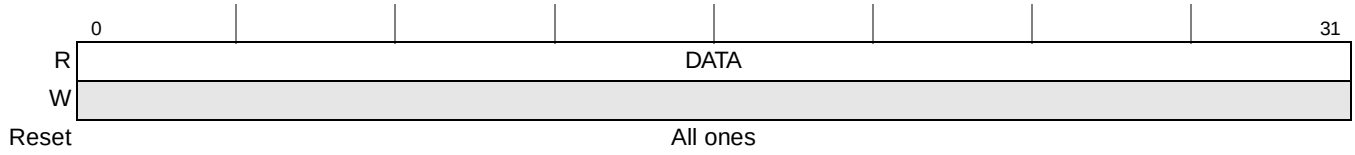


Figure 21-23. SPI Receive Data Register (SPIRD)

21.9.7.1 SPIRD Examples with Some SPMODE[REV,LEN] Values

Access: Read Only

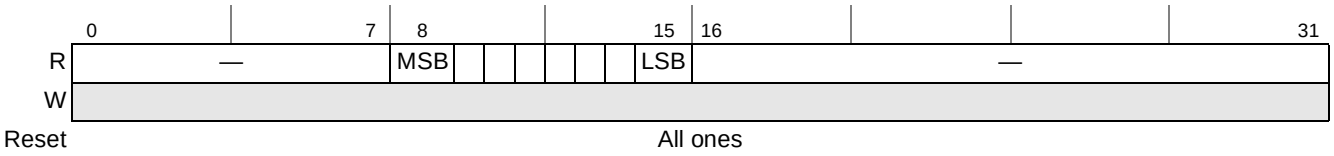


Figure 21-24. SPIRD in REV=1 LEN=0x7

Access: Read Only

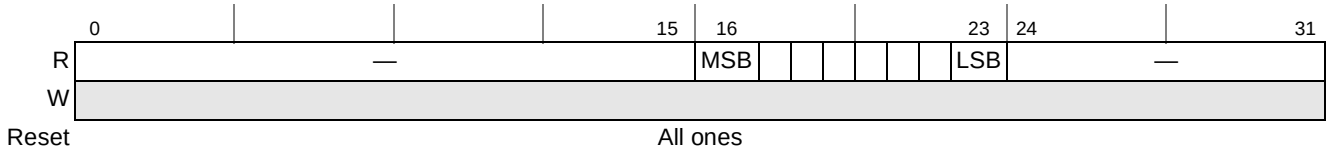


Figure 21-25. SPIRD in REV=0 LEN=0x7

Access: Read Only

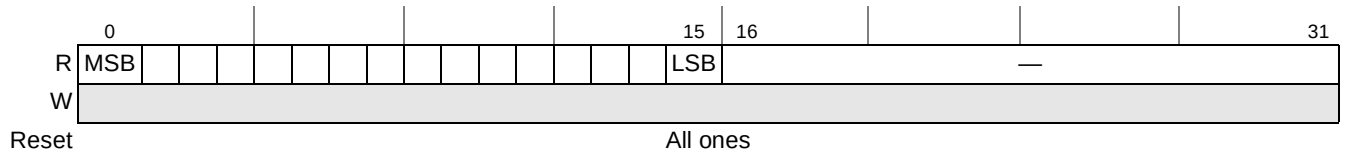


Figure 21-26. SPIRD in REV=1 LEN=0xF

Access: Read Only

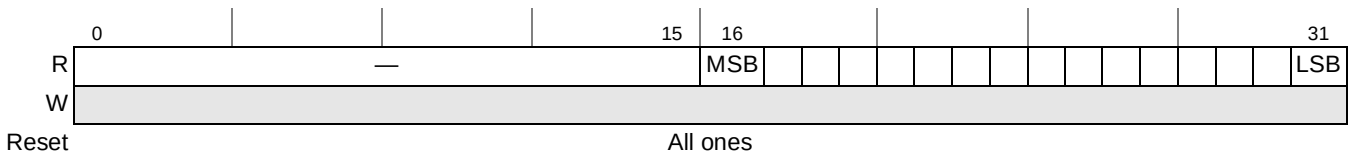


Figure 21-27. SPIRD in REV=0 LEN=0xF

Chapter 22
Unified Communications Controllers (UCCs)

This chapter provides a general overview of the UCCs, the set of the protocols for them, and the common UCC programming model. For details on the feature set and the protocol-specific programming model, refer to subsequent UCC chapters.

The QUICC Engine block UCC implement a wide range of protocols and interfaces. The supported protocols are Ethernet, UART, BISYNC, HDLC, Transparent, ATM, Serial ATM, and QMC. The supported interfaces include RS-232, MII/RMII and UTOPIA L2. Each UCC can also be connected to the TSA in HDLC, Transparent, QMC, or Serial ATM.

22.1 Overview

The combined UCC hardware and RISC firmware provide an excellent platform for the efficient implementation of a large variety of protocols at various levels of the seven-layer OSI model. Together, they can provide functions such as termination, bridging, switching, routing, and interacting to interface with a wide variety of standard WANs, LANs, and proprietary networks.

The QUICC Engine block can be configured to implement different protocols concurrently by configuring each UCC to run a different protocol as needed by the target application.

The UCC is a unification of the legacy peripherals found in the first generations of the PowerQUICC family of devices: the serial communication controller (SCC) and the fast communication controller (FCC). The unification enhances function and performance. For example, the UCC allows the user to program its FIFO size at initialization, which supports optimized allocation of the memory space for slow and fast protocols. The range of line speeds supported may vary from a few Kbps (for UART) to 100Mbps (100 BaseT Ethernet) full duplex.

The protocols in the UCC are split into two categories:

- Slow Protocols:
  - UART
  - BISYNC
  - QMC
  - Serial ATM
- Fast Protocols:
  - HDLC
  - Transparent
  - 10/100 BaseT Ethernet
  - ATM

Subsequent chapters in this book describe these protocols and their extended features.

The UCC programming model is similar to the programming model of the SCC and FCC in the MPC82xx family of devices. In the ATM and Ethernet, the initialization of the data structures differs from the MPC82xx. Figure 22-1 shows the UCC block diagram.

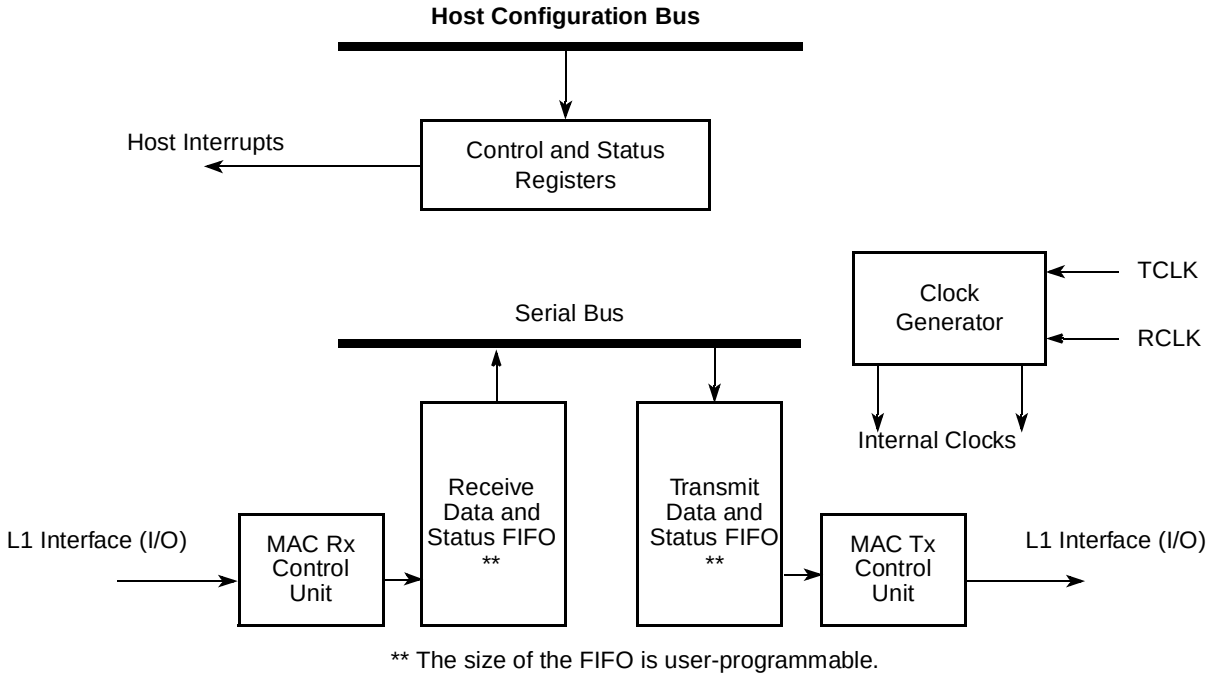


Figure 22-1. UCC Block Diagram

22.2 UCC Feature Set

The UCC feature set includes:

- HDLC/SDLC, HDLC bus, and transparent in single and nibble bit modes
- 10/100 802.3 Ethernet through MII, RMII interfaces
- ATM up to OC-3 (155 Mbps) rate AAL5 through UTOPIA L2 8--bit interface
- Optionally connected to the external TDM interfaces through the TSA
  - Serial or nibble data port size
- UCC clocks can be derived from a baud-rate generator or from an external clock source
- Supports $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem control signals
- Uses burst accesses to external memory for HDLC, HDLC bus, transparent, Ethernet, ATM
- Uses single accesses to memory in UART, BISYNC, and QMC modes
- Relocatable multibuffer data structures for receive and transmit with buffer descriptors
- External BDs for Ethernet, HDLC, HDLC bus, transparent, and ATM protocols
- Internal BDs for UART, BISYNC, and QMC modes
- 82xx Family of RAM-based microcode available

- User-programmable FIFO size
- Full duplex operation
- Echo and local loopback modes for testing
- Features that are not supported in the QUICC Engine module and were supported in the MPC82xx CPM are:
  - DPLL
  - AppleTalk
  - Serial Ethernet interface

The internal clocks (RCLK, TCLK) for each UCC can be programmed to use either an external or internal source. The internal clocks originate from one of the baud-rate generators. The rate of these clocks can be up to one-half of the QUICC Engine clock frequency. However, the UCC's ability to support a sustained bit stream depends on the protocol and other factors.

Each UCC can be connected to its own set of pins on the CE. This configuration is called non-multiplexed serial interface, or NMSI. Optionally, multiple UCCs can be connected to a single TDM interface through the serial interface (SI). Each UCC is connected to the interrupt controller as a level-interrupt source with programmable priority. In the NMSI configuration, each UCC can support the standard modem interface signals (RTS, CTS, and CD) through the appropriate port pins and the interrupt controller. Additional handshake signals can be supported with additional parallel I/O lines.

22.2.1 UCC Base Addresses

22.2.1.1 UCC Page Base Address

The QUICC Engine module maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the UCCs. UCC default base addresses are described in Table 22-1.

The UCC Parameter RAM base addresses are programmable using the ASSIGN PAGE host command. See Section 19.3.1.1.1, "assign page Command," for details.

NOTES: Backward Compatibility

UCC1-UCC3 are backward-compatible to FCC1-FCC3, and UCC5 is backward-compatible to SCC1 in both their parameter RAM base address, and in their respective protocols (UCC1-UCC4 are set for fast protocols, and UCC5 is set for slow protocols).

In the 82xx family of devices, the parameter RAM for each serial controller is located at a fixed address in the internal memory space for the device. The QUICC Engine architecture, however, has a default setting for the parameter RAM pages (see Table 22-1), but the user must relocate the pages to a location within the RAM space.

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 from the HDLC parameter RAM.

Table 22-1. Parameter RAM—UCC Default Base Addresses

Page	Address [1]	Peripheral	Size (Bytes)
1	0x8400	UCC1	256
2	0x8500	UCC2	256
3	0x8600	UCC3	256
4	0x9000	UCC4	256
5	0x8000	UCC5	256

1 Offset from RAM_Base. In the QUICC Engine module the default address is outside valid RAM space and the user must relocate the page.

22.2.1.2 UCC Registers Base Addresses

The UCCs registers base address, as an offset from the QUICC Engine internal memory map base address, are listed in Table 22-2.

Table 22-2. UCC Register Base Addresses

Peripheral	Address[1]	Size (Bytes)
UCC1	0x2000	512
UCC2	0x3000	512
UCC3	0x2200	512
UCC4	0x3200	512
UCC5	0x2400	512

1 Offset from QUICC Engine module Base

22.3 Programming Model

22.3.1 Registers Overview

Before the UCC is configured for protocol-specific operation, it should be set to either fast or slow protocols mode through the general UCC extended mode register (GUEMR). This should be done as the first stage of the initialization process. For slow protocols, see Chapter 23, “UCC for Slow Protocols.” For fast protocols, see Chapter 26, “UCC for Fast Protocols.”

Table 22-3. UCC Registers

Address[1] Fast Protocol	Address Slow Protocol	Name	Size (Bits)	Access
0x90		General UCC extended mode register (GUEMR)	8	R/W
0x3C		UCC transmit polling timer (UTPT)	16	R/W
0x8	0xC	UCC transmit on demand register (UTODR)	16	R/W
0xC	0xE	UCC data synchronization register (UDSR)	16	R/W

Table 22-3. UCC Registers (continued)

Address[1] Fast Protocol	Address Slow Protocol	Name	Size (Bits)	Access
0x10	—	UCC event register, fast (UCCE)	32	R/W
—	0x10	UCC event register, slow (UCCE)	16	R/W
0x14	—	UCC mask register, fast (UCCM)	32	R/W
—	0x14	UCC mask register, slow (UCCM)	16	R/W
0x18	0x17	UCC status register (UCCS)	16	R/W

Note:

1 Offset from UCCx base.

22.3.2 General UCC Extended Mode Register (GUEMR)

GUEMR, shown in Figure 22-2, configures the UCC to operate as a fast or a slow communication controller. Also, it has ATM protocol-specific mode settings. For these settings, refer to Section 26.4.2.1, "GUMR in Fast Mode."

Note that while it is possible to program the UCC receiver and transmitter functions independently, they should always be set to the same configuration (that is, for fast or slow protocols).

Address: UCC*x*_Base + 0x90 Access: Read/Write

	0			3		5	6	7
R W	—						URMODE	UTMODE
Reset	0	0	0	1	0	0	*n*[1]	*n*[1]

Figure 22-2. General UCC Extended Mode Register

1 See Table 22-5.

Table 22-4 describes the GUEMR fields.

Table 22-4. GUEMR Field Descriptions

Field	Name	Description
0–2	Reserved	Must be cleared
3	Reserved	Must be set
4–5	Reserved	Must be cleared
6	URMODE	UCC Rx Mode 0 UCC Rx is configured for Slow protocols 1 UCC Rx is configured for Fast protocols **Note:** URMODE should be programmed to have the same value as the UTMODE bit.
7	UTMODE	UCC Tx Mode 0 UCC Tx is configured for Slow protocols 1 UCC Tx is configured for Fast protocols

Table 22-5. GUEMRx Reset Value

UCC	Reset Value	Description
UCC1	0x13	Fast protocols
UCC2	0x13	Fast protocols
UCC3	0x13	Fast protocols
UCC4	0x13	Fast protocols
UCC5	0x10	Slow protocols

22.3.3 UCC Transmit Polling Timer (UTPT)

The UTPT, shown in Figure 22-3, configures the amount of time in serial clocks between consecutive polls of an empty transmit BD. This register gets the value of 256 clocks during reset (for backward compatibility). Program it only if a different value is necessary.

Address: UCCx_Base + 0x3C Access: Read/Write

	0							7								15
R	MCNT															
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 22-3. UCC Transmit Polling Timer

The UTPT register is described in Table 22-6.

Table 22-6. UTPT Register Field Description

Bit	Name	Description
0–15	MCNT	Transmit polling max count (in serial clock cycles). When the UCC polls for a ready Transmit BD, this is the time interval (in serial clock cycles) between polling. The default and recommended value (for most applications) is 256. The minimum value for MCNT is 64. **Note:** MCNT has an impact on the entire QUICC Engine block. It is recommended to keep the default value unless required by application.

22.3.4 UCC Transmit-On-Demand Register (UTODR)

In normal operation, if no frame is being sent by the UCC, the QUICC Engine module periodically polls the Ready bit of the next TxBD to see if the user requested transmission of a new frame/buffer. The polling algorithm depends on the state of the UCC, but occurs every parametric number of serial transmit clocks programmed in the UFPT register (in the MCNT bit field). However, the user can request that the QUICC Engine module should 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 (UTODR) after setting TxBD[R] of the frame/buffer. UTODR is shown in Figure 22-4.

In LAN-type protocols where the protocol specification limits maximum interframe GAP times, the transmit-on-demand feature can be used to give a high priority to a specific TxBD.

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, and transmit-on-demand is not required.

Address: Fast protocols: UCC BASE + 0x08
Slow protocols: UCC BASE + 0xC

Access: Read/Write

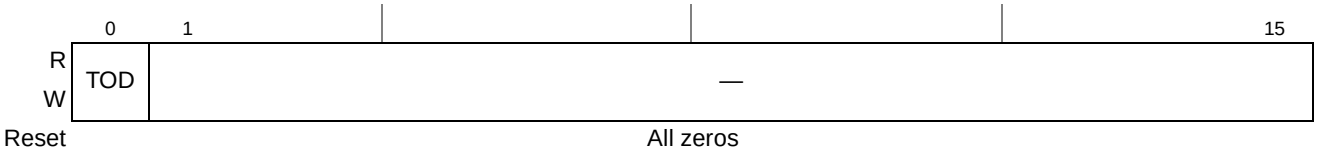


Figure 22-4. UCC Transmit-on-Demand Register

Fields in the UTODR are described in Table 22-7.

Table 22-7. UTODR Field Descriptions

Field	Name	Description
0	TOD	Transmit on demand 0 Normal polling 1 The QUICC Engine module gives high priority to the current TxBD and begins sending the frame without waiting for the normal polling time to check TxBD[R]. TOD is cleared automatically.
1–15	—	Reserved. Should be cleared.

22.3.5 UCC Event Register (UCCE)

Each UCC has a 32-bit event register (UCCE) used to report events. Some protocols only use the higher 16-bits of the event register. When an event is recognized, the UCC sets its corresponding UCCE 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. UCCE is cleared at reset. Fields of this register are protocol-dependent and are described in the respective protocol sections.

22.3.6 UCC Mask Register (UCCM)

Each UCC has a 32-bit read/write UCC mask register (UCCM) that enables or disables QUICC Engine interrupts to the system for events reported in an event register (UCCE). Some protocols only use the higher 16-bit of the event register. Bit positions in UCCM are identical to those in UCCE. Note that an interrupt is generated only if the UCC interrupts are also enabled in the QUICC Engine interrupt controller; see Section 18.3.11, “QUICC Engine System Interrupt Mask Register (CIMR).”

If a UCCM bit is zero, the QUICC Engine module does not proceed with its usual interrupt handling whenever that event occurs. Whenever a bit in the UCCM register is set (such as, equal to 1), a 1 in the corresponding bit in the UCCE register sets the UCC event bit in the interrupt pending register. See QUICC Engine System Interrupt Pending Register in Section 18.3.10, “QUICC Engine System Interrupt Pending Register (CIPNR).”

22.3.7 UCC Status Register

Each UCC has an 8-bit, read/write UCC status register (UCCS) that lets the user monitor real-time status conditions (such as flags or idle) on the RxD line. See details about this register in the protocol chapters (HDLC, Transparent, UART).

22.4 Handling UCC Interrupts

To allow interrupt handling for UCC-specific QUICC Engine block events, Event-, Mask-, and Status-registers are provided within each UCC's internal memory map area (see Table 22-8). Because interrupt events are protocol-dependent, event descriptions are found in the specific protocol chapters. The UCC has 32-bit event and mask registers, but only the Ethernet protocol uses 32-bits, the rest of the protocols uses the16-bit event and mask registers.

Table 22-8. UCC*x* Event, Mask, and Status Registers

Registers	Description
UCCE*x*	UCC event register (high and low). This 32-bit register reports events recognized by any of the UCCs. When an event is recognized, the UCC sets its corresponding bit in UCCE, regardless of the corresponding mask bit. When the corresponding event occurs, an interrupt is signaled to the QUICC Engine interrupt controller. Bits are cleared by writing ones (writing zeros has no effect). UCCE is cleared at reset and can be read at any time.
UCCM*x*	UCC mask register (high and low). The 32-bit read/write register allows interrupts to be enabled or disabled using the QUICC Engine module for specific events in each UCC channel. An interrupt is generated only if UCC interrupts in this channel are enabled in the QUICC Engine interrupt mask register (CIMR). If a UCCM bit is zero, the QUICC Engine module does not proceed with interrupt handling when that event occurs. The UCCM and UCCE bit positions are identical.
UCCS*x*	UCC status register. This 8-bit, read-only register allows monitoring of the real-time status of RXD.

Follow these steps to handle a UCC interrupt:

1. When an interrupt occurs, read UCCE to determine the interrupt sources and clear those UCCE bits (in most cases). Bits are cleared by writing ones (writing zeros has no effect).
2. Process the TxBDs to reuse them if UCCE[TX] or UCCE[TXE] = 1. If the transmit speed is fast or the interrupt delay is long, the UCC 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 UCCE[RX], UCCE[RXB], or UCCE[RXF] is set. As with transmit buffers, if the receive speed is fast or the interrupt delay is long, the UCC 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.

Additional information about interrupt handling can be found in Section 18.2, "Interrupt Controller."

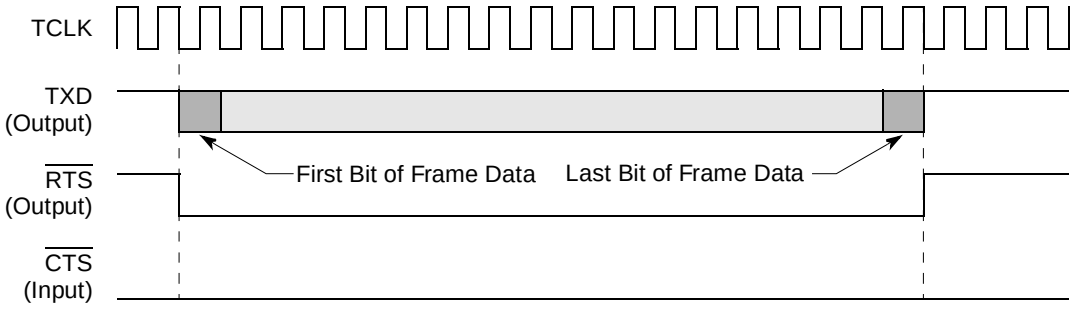
22.5 Controlling UCC Timing with $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$

When the UCC is programmed to normal operation (non loopback) in NMSI mode (not through the TSA), external hardware flow control signals can be used. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are controls external to the UCC. In the following subsections, a normal transmit clock operation is assumed, implying a non-inverted Tx clock.

22.5.1 Synchronous Protocols

$\overline{\text{RTS}}$ is asserted when the UCC data is loaded into the Tx FIFO and a falling Tx clock occurs. At this point, the UCC 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 22-5 shows that the delay between $\overline{\text{RTS}}$ and data is 0 bit cycles, regardless of the status of CTSS. This operation assumes that $\overline{\text{CTS}}$ is already asserted to the UCC or that $\overline{\text{CTS}}$ is reprogrammed to be a general purpose parallel I/O line, in which case $\overline{\text{CTS}}$ to the UCC 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 22-5. 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 22-6 shows that the delay between $\overline{\text{CTS}}$ and the data can be approximately 0.5 to 1 bit cycles or no delay, depending on the status of CTSS.

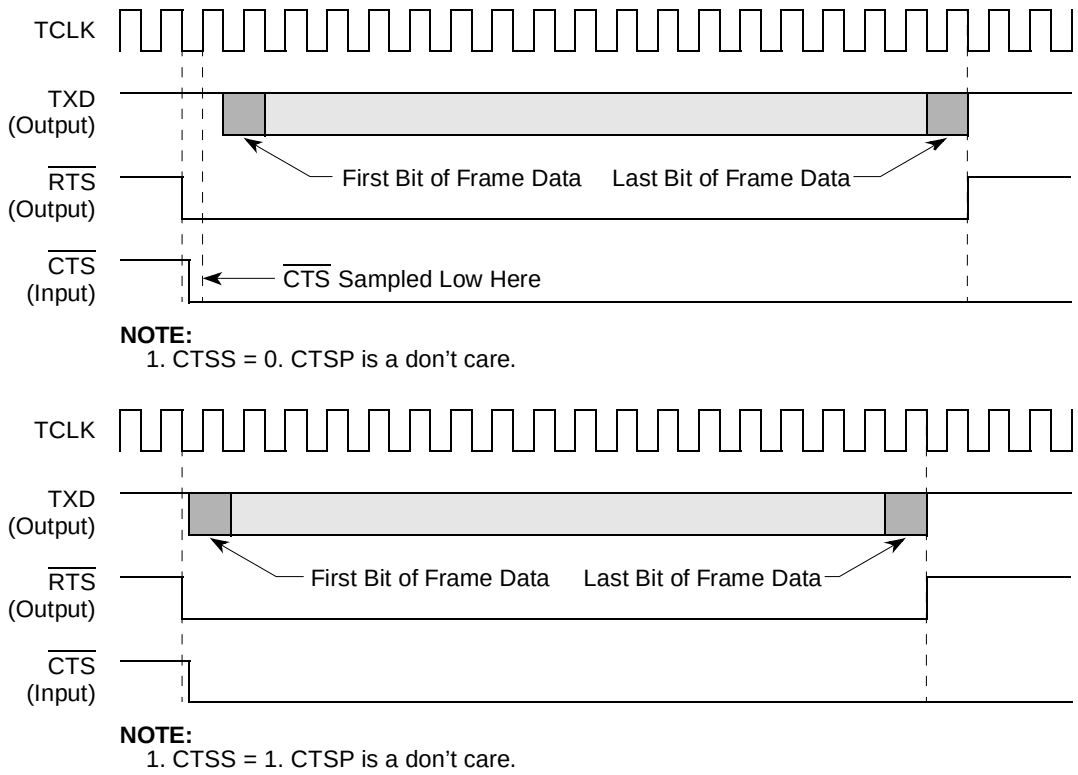


Figure 22-6. Output Delay from $\overline{\text{CTS}}$ Asserted for Synchronous Protocols

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 CTSS is zero, the UCC 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 22-7.

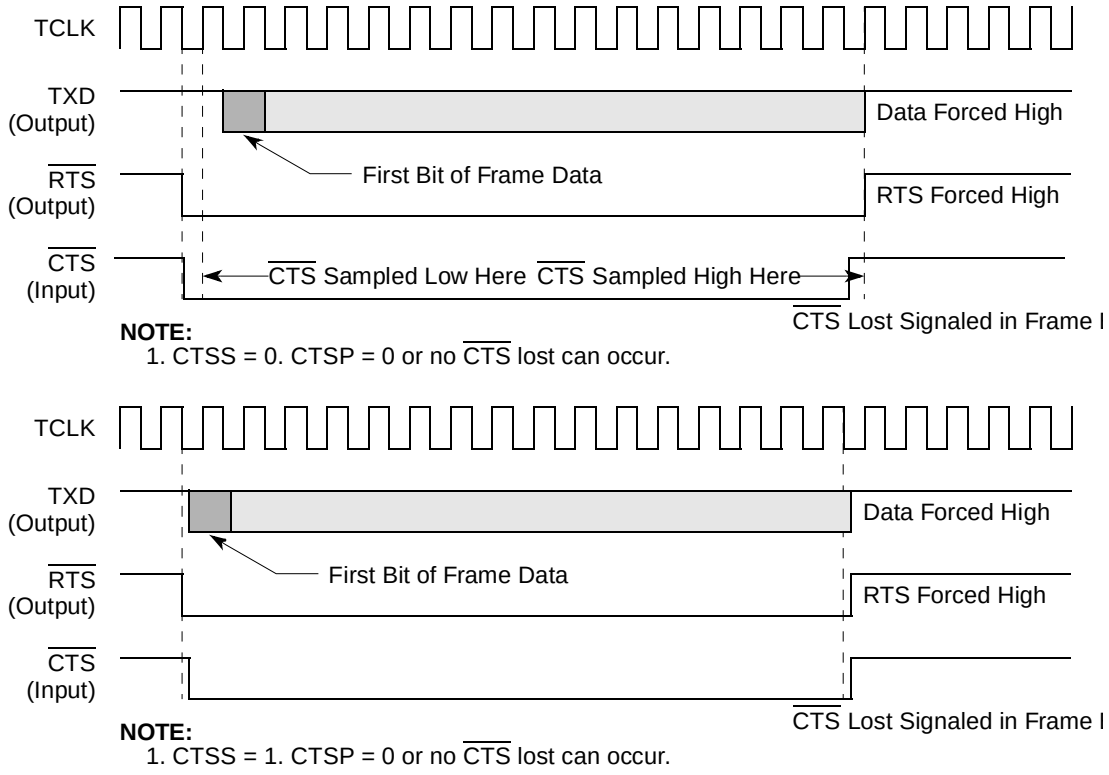
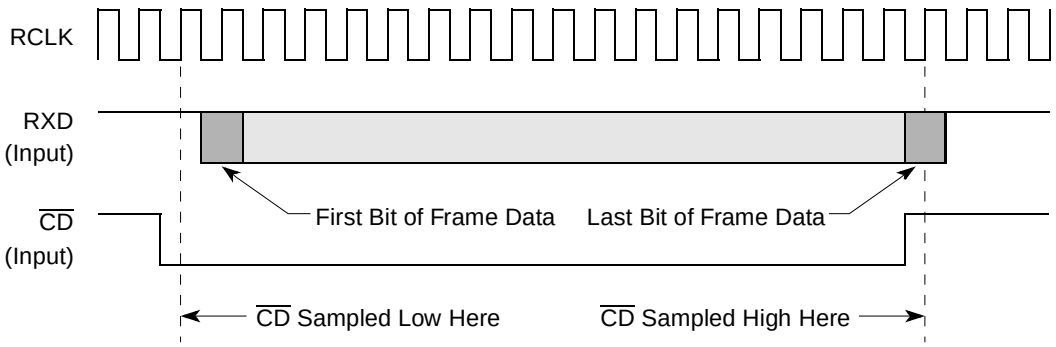


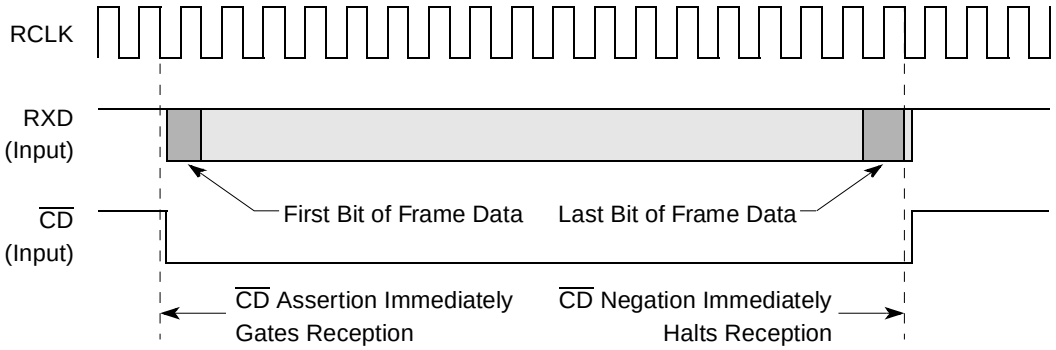
Figure 22-7. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

Note that if CTSS = 1, $\overline{\text{CTS}}$ transitions must occur while the Tx clock is low.

Reception delays are determined by $\overline{CD}$ as shown in Figure 22-8. If CDS is zero, $\overline{CD}$ is sampled on the rising Rx clock edge before data is received. If CDS is 1, $\overline{CD}$ transitions cause data to be immediately gated into the receiver.



NOTES:
1. 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 frame BD.
3. If CDP = 1, $\overline{CD}$ lost cannot occur and $\overline{CD}$ negation has no effect on reception.



NOTES:
1. 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 frame BD.
3. If CDP = 1, $\overline{CD}$ lost cannot occur and $\overline{CD}$ negation has no effect on reception.

Figure 22-8. 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 CDS is cleared, the UCC must sample $\overline{CD}$ before a $\overline{CD}$ lost error is recognized; otherwise, the negation of $\overline{CD}$ immediately causes the $\overline{CD}$ lost condition. If CDS is set, all $\overline{CD}$ transitions must occur while the Rx clock is low.

22.5.2 Asynchronous Protocols

In asynchronous protocols, $\overline{RTS}$ is asserted when UCC 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 24, "UART Mode and Asynchronous HDLC."

- If $\overline{\text{CTS}}$ is already asserted when $\overline{\text{RTS}}$ is asserted, transmission begins in two additional bit times.
- If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ is asserted and CTSS = 0, transmission begins after three additional bit cycles have elapsed.
- If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ is asserted and CTSS = 1, transmission begins after two additional bit cycles have elapsed.

22.5.3 Data Encoding

The UCC can be programmed to encode and decode the UCC data as NRZ, NRZI Mark, and NRZI Space. It can be programmed to invert the data stream of a transfer in all encodings, including the standard NRZ format. Also, when the transmitter is idling, the UCC can either force TXD high or continue encoding the data supplied to it. Figure 22-9 shows the different encoding methods.

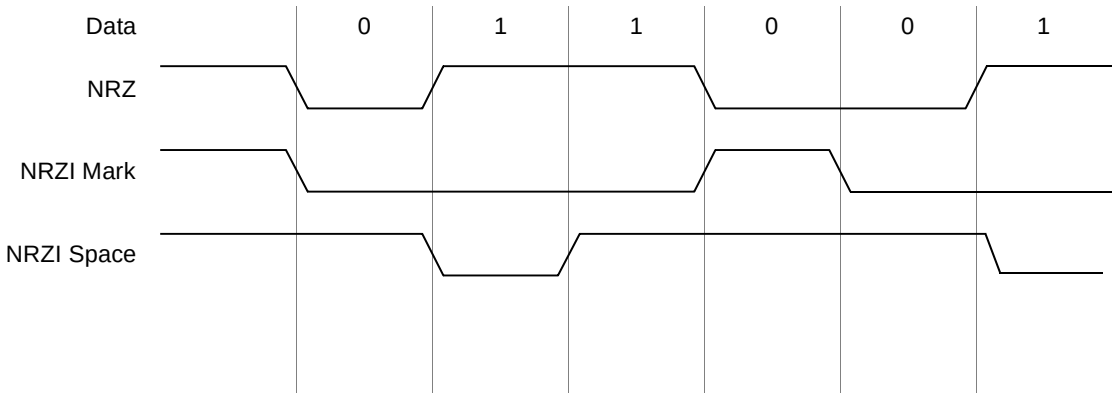


Figure 22-9. Data Encoding Examples

NRZ or NRZI codings can be selected in RENC/TENC and RINV/TINV. Coding definitions are shown in Table 22-9.

Table 22-9. Data 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[1]	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.

Note:

1 Available only for slow protocols.

22.6 UCC Initialization Sequence

NOTE

GUEMR must be the first register to be programmed in the UCC initialization sequence.

The UCC basic mode is configured by four primary fields:

- GUEMR[URMODE], GUEMR[UTMODE].
- Fast protocols: GUMR[MODE], GUMR[TTX], GUMR[TRX] (Section 26.4.2, "UCC Registers in Fast Mode").
- Slow protocols: GUMR_L[MODE], GUMR_H[TTX], GUMR_H[TRX] (Section 23.2.2, "General UCC Mode Registers (GUMR)").

Because there is a different memory map for fast and slow protocols, the first register programmed is GUEMR, which configures the UCC to either Fast or Slow modes. See Section 22.3.2, "General UCC Extended Mode Register (GUEMR)." In most cases, URMODE should be the same as UTMODE. One half of the UCC can be used as a transparent controller while the other half is set to the HDLC protocol.

Table 22-10. Initialization Options

Protocol	UTMODE[1]	URMODE	Type	MODE[2]	TTX[3]	TRX[3]
ATM	1	1	Fast	1010	0	0
Ethernet	1	1	Fast	1100	0	0
HDLC	1	1	Fast	0000	0	0
Transparent	1	1	Fast	0000	1	1
Tx Transparent, Rx HDLC	1	1	Fast	0000	1	0
Tx HDLC, Rx Transparent	1	1	Fast	0000	0	1
QMC, Serial ATM	0	0	Slow	0010	1	1
UART	0	0	Slow	0100	0	0
Async HDLC	0	0	Slow	0110	0	0
BISYNC	0	0	Slow	1000	0	0

Note:

1 Located in the GUEMR

2 GUMR[27:31] or GUMR_L[27:31], both are at the same address offset 0x0 from UCC Base.

3 For fast protocols TTX and TRX are set in GUMR register. For slow protocols TTX and TRX are set in GUMR_H register.

22.7 UCC Common Initialization Sequence

The UCC requires a number of registers and parameters to be configured after power-on reset. The following sequence is common to protocols.

1. If the TSA is used (for QMC, Clear Channel TDM, ISDN IDL or other applications), the SI must be configured.
2. If the UCC is routed to the UPC (for ATM), the UPC must be configured.
3. Write to the I/O port configuration registers to configure and connect the I/O pins to the UCC.
4. Initialize the QUICC Engine multiplexing logic (clock routing and SI/TSA routing).
5. Write to GUEMR[UTMODE] and GUEMR[URMODE]
6. Optional: Initialize the Parameter RAM page offset for the UCC.
7. Clear out any current events in UCCE, as needed.
8. Write the UCCM register to enable the interrupts in the UCCE register.
9. Clear out any current interrupts in the CIPNR, if preferred.
10. Write the CIMR to enable interrupts to the on-chip interrupt controller.
11. Proceed with UCC initialization sequence for fast or slow protocols.

Chapter 23
UCC for Slow Protocols

A UCC can be programmed to provide the same support found on MPC82xx PowerQUICC II devices' serial communications controllers (SCCs). For the QUICC Engine module, this functionality is referred to as "slow". When programmed as a slow communication controller, a UCC can be used for UART, BISYNC, or multi-channel HDLC (QMC) protocols. When configuring a UCC to one of these protocols, read this chapter first and then proceed to the protocol specific chapter:

- Chapter 24, "UART Mode and Asynchronous HDLC"
- Chapter 25, "BISYNC Mode"
- Chapter 34, "QUICC Multi-Channel Controller (QMC)"

NOTE

In MPC832x the UART, Asynchronous HDLC and BISYNC protocols requires a software patch from Freescale. Please contact an FAE.

The UCC is backward compatible with the PowerQUICC II serial communication controller (SCC), except for the following features which are not supported:

- DPLL (only NRZ and NRZI data encoding are supported)
- AppleTalk
- Serial Ethernet interface
- SS7

A UCC can be connected to its own set of pins. This configuration is called the non-multiplexed serial interface (NMSI). Using NMSI, a UCC 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 provide additional handshake signals.

An alternate configuration is when the UCC is connected to a TDM interface through the TSA, as required for the QMC protocol.

23.1 Features

The following is a list of the main UCC features.

- Implements synchronous start/stop, asynchronous start/stop (UART)
- Clocks can be derived from a baud rate generator (internal to the QUICC Engine module) or from an external pin
- Data rate for asynchronous communication can be as high as 1/16 of the QUICC Engine clock frequency
- Supports automatic control of the $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem signals

- Multi-buffer data structure for receiving and transmitting (the number of buffer descriptors ((BDs)) is limited only by the size of the internal multi-user RAM—8 bytes per BD)
- Transmit-on-demand feature decreases time to frame transmission (transmit latency)
- Low FIFO latency option for transmitting and receiving in character-oriented protocols
- Full-duplex operation
- Echo and local loopback modes for testing

23.2 Programming Model

The following sections discuss the programming model, and registers GUMR_L, GUMR_H which are common to all protocols. The protocol implemented by a UCC is selected by the value in the GUMR_L[MODE] bit field. Each UCC has an additional protocol-specific mode register (PSMR) that configures it for the chosen protocol. The PSMR fields are described in the protocol-specific chapters.

23.2.1 UCC Memory Map for Slow Protocols

Table 23-1. UCC Memory Map (Slow Protocols)

Address[1]	Use	Size (bits)	Access
0x0	General UCC Mode Register (GUMR_L)	32	R/W
0x4	General UCC Mode Register (GUMR_H)	32	R/W
0x8	Protocol Specific Mode Register (PSMR)	16	R/W
0xC	UCC Transmit On Demand Register (UTODR)	16	R/W
0xE	UCC Data Synchronization Register (UDSR)	16	R/W
0x10	UCC Event Register (UCCE)	16	R/W
0x14	UCC Mask Register (UCCM)	16	R/W
0x17	UCC Status Register (UCCS)	8	R
0x3C	UCC Transmit Polling Timer (UTPT)	16	R/W
0x90	General UCC Extended Mode Register (GUEMR)	8	R/W

1 Offset from UCCx base.

23.2.2 General UCC Mode Registers (GUMR)

Each UCC contains a GUMR that defines options common to most of the protocols. GUMR_L, shown in Figure 23-1, contains the low-order 32 bits; GUMR_H, shown in Figure 23-2, contains the high-order 32 bits. Some GUMR operations are described in later sections.

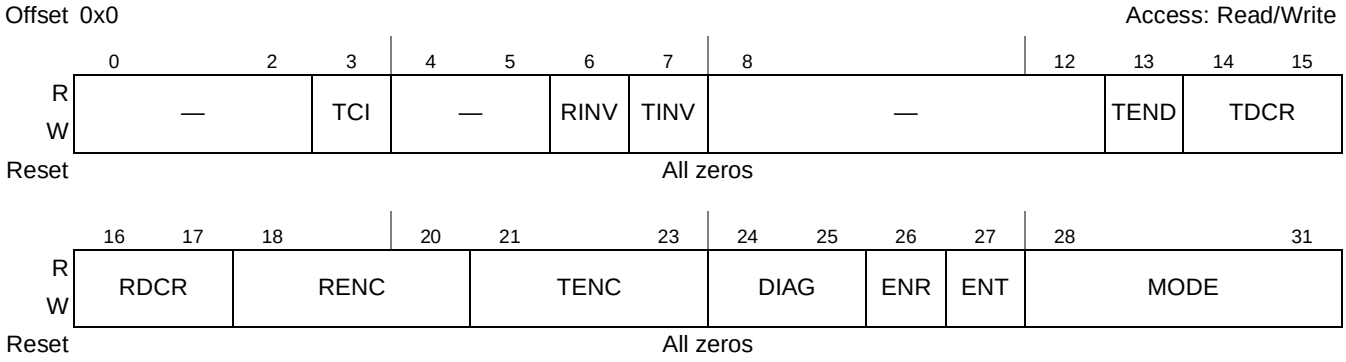


Figure 23-1. GUMR_L—General UCC Mode Register Low Order

Table 23-2 shows the field descriptions for GUMR_L.

Table 23-2. GUMR_L Field Descriptions

Bit	Name	Description
0–2	—	Reserved, should be cleared.
3	TCI	Transmit clock invert. 0 Normal operation. 1 Before it is used, the internal Tx clock (TCLK) is inverted by the UCC so it can clock data out one-half clock earlier (on the rising rather than the falling edge). In this case, the UCC offers a minimum and maximum rising clock edge-to-data specification. Data output by the UCC 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. **Note:** Applies only for the BISYNC protocol. **Note:** This bit should be cleared if this UCC is used with the TSA.
4–5	—	Reserved, should be cleared
6	RINV	Rx invert Data. Must be zero in asynchronous UART mode. 0 Do not invert. 1 Invert data before sending it for the UCC receiver. Used to produce NRZI space from NRZI mark or to invert data stream in regular NRZ mode.
7	TINV	Tx invert Data. 0 Do not invert. 1 Invert data before sending to the UCC encoder. Used to produce NRZI space from NRZI mark or to invert data stream in regular NRZ mode.
8–12	—	Reserved, should be cleared
13	TEND	Transmitter frame ending. Intended for NRZI transmitter encoding. TEND determines whether TXD should be idle in a high state or in encoded ones state (high or low). 0 Default operation. TXD is encoded only when data is sent. 1 TXD is always encoded, even when idles are sent.

Table 23-2. GUMR_L Field Descriptions (continued)

Bit	Name	Description
14–15	TDCR	Transmitter/Receiver oversampling rate. TDCR should match RDCR in most applications to allow the transmitter and receiver to use the same clock source. 00 1x clock mode. Default operation 01 8x clock mode 10 16x clock mode. Normally chosen for UART 11 32x clock mode
16–17	RDCR	
18–20	RENC	Receiver decoding/transmitter encoding method. RENC should equal TENC in most applications. 000 NRZ (default setting). Required for UART (synchronous or asynchronous). 001 NRZI Mark (set RINV/TINV also for NRZI space). 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 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 UCC uses modem signals to enable or disable transmission and reception. Timing diagrams for these modem signals are shown in Section 22.5, “Controlling UCC Timing with RTS, CTS, and CD” 01 Local loopback mode. The 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 UCC 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 32.6.3, “SI Mode Register (SIxMR).”
26	ENR	Enable receive. Enables the receiver hardware state machine for this UCC. 0 Reset the UCC receiver. The receiver is disabled and data in the Rx FIFO is lost. 1 The receiver is enabled. Section 23.4.6, “Reconfiguring the UCC” describes how to disable/enable a UCC. Note that other tools, such as the ENTER HUNT MODE command, and the E bit of the RxBD, also provide capability to control the receiver.
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this UCC. 0 Reset the UCC transmitter. 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 TxFIFOs is flushed. 1 The transmitter is enabled. Section 23.4.6, “Reconfiguring the UCC” describes how to disable/enable a UCC. 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.

Table 23-2. GUMR_L Field Descriptions (continued)

Bit	Name	Description
28–31	MODE	Channel protocol mode. 0010 QMC or Serial ATM 0100 UART 0110 Async HDLC (this is based on UART protocol see Chapter 24, “UART Mode and Asynchronous HDLC”) 1000 BISYNC All other values are reserved

Figure 23-2 shows the GUMR_H register.

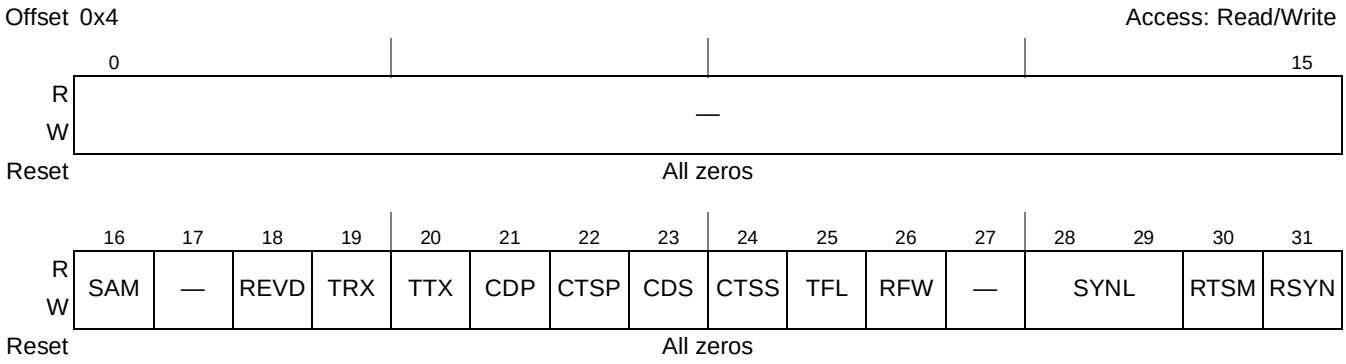


Figure 23-2. GUMR_H—General UCC Mode Register (High Order)

Table 23-3 describes GUMR_H fields.

Table 23-3. GUMR_H Field Descriptions

Bit	Name	Description
0–15	—	Reserved, should be cleared.
16	SAM	Serial ATM (Extension to GUMR_L[MODE] for QMC/Serial ATM) 0 QMC 1 Serial ATM
17	—	Reserved, should be cleared.
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 UCC (either the receiver, transmitter, or both) and sends the msb of each byte first. Section 25.3.2.5, “BISYNC Mode Register (UPSMR)” describes reversing bit order in a BISYNC protocol.
19–20	TRX, TTX	Transparent receiver/transmitter. Set these bits for QMC protocol. Clear these bits for other slow protocols. 0 Normal operation (use in non QMC/SAM protocols) 1 The channel uses transparent mode, should be used in QMC/SAM (HDLC, transparent or Serial ATM) protocol.

Table 23-3. GUMR_H Field Descriptions (continued)

Bit	Name	Description
21, 22	CDP, CTSP	$\overline{CD}$/$\overline{CTS}$ pulse. If this UCC is used in the TSA and is programmed in transparent mode, set CTSP and refer to Section 28.5.3.2, "Synchronization and the TSA for options to program 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. **Note**: CTSP should be cleared in internal loopback.
23, 24	CDS, CTSS	$\overline{CD}$/$\overline{CTS}$ sampling. Determine synchronization characteristics of $\overline{CD}$ and $\overline{CTS}$. If the UCC 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 UCC, and 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, when the transfer begins. Useful for two communication processors in transparent mode because the $\overline{RTS}$ of one device can connect directly to the $\overline{CD}$/$\overline{CTS}$ of another. **Note**: CDS Should be set in internal loopback
25	TFL	Transmit FIFO length. 0 Normal operation. The transmit FIFO is 128 bytes. 1 The Tx FIFO is 1 word. 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. It is 48 bytes total. Data is not normally written to receive buffers until at least 32 bits are received. This configuration is required for QMC protocols. 1 Low-latency operation. The receive FIFO is 8 bits wide, reducing the Rx FIFO to a quarter of 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 (except if $\overline{CD}$ signal is not used for flow control). **Note**: Should be set in BISYNC and UART protocols.
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 UCC. Assuming $\overline{CTS}$ is asserted, transmission begins 8 clocks after the receiver starts receiving data. **Note**: This bit should be cleared in all protocols except for transparent operation. Can be set only in a serial (1 bit data width) interface
28–29	SYNL	Sync length (BISYNC and transparent mode only). See the data synchronization register (UDSR) definition in Section 23.2.3, "UCC Data Synchronization Register (UDSR).". 00 An external sync ($\overline{CD}$) is used instead of the sync pattern in the UDSR. 01 4-bit sync. The receiver synchronizes on a 4-bit sync pattern stored in the UDSR. This sync and additional syncs can be stripped by programming the UCC'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 UDSR. 11 16-bit sync. Also called BISYNC. The receiver synchronizes on a 16-bit sync pattern stored in the UDSR.

Table 23-3. GUMR_H Field Descriptions (continued)

Bit	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 UCC is enabled.
31	RSYN	Receive synchronization timing (totally transparent mode only). 0 Normal operation or internal loopback. 1 If CDS = 1, $\overline{\text{CD}}$ should be asserted on the second bit of the Rx frame rather than on the first. **Note**: Should be cleared in all protocols except for transparent operation. Can be set only in a serial (1 bit data width) interface. **Note:** This bit must be cleared if this UCC is used in internal loopback.

23.2.3 UCC Data Synchronization Register (UDSR)

Each UCC has a UDSR that specifies the pattern used for frame synchronization. The programmed value for UDSR depends on the protocol:

- UART — UDSR is used to configure fractional stop bit transmission.
- BISYNC — UDSR should be programmed with the sync pattern.

Figure 23-3 shows the UDSR fields.

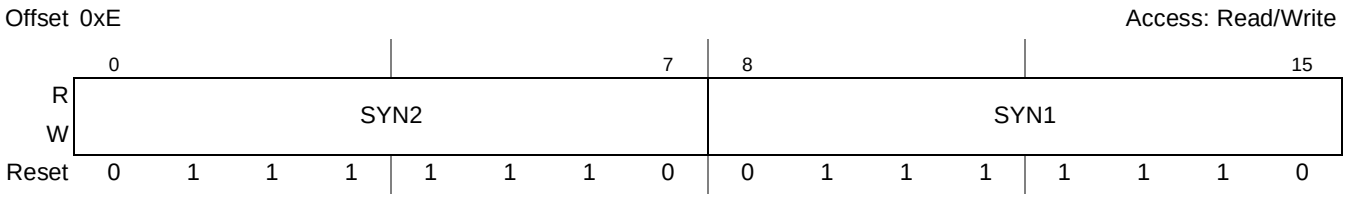


Figure 23-3. Data Synchronization Register (UDSR)

23.2.4 UCC Transmit Polling Timer (UTPT)

Refer to Section 22.3.3, “UCC Transmit Polling Timer (UTPT).”

23.2.5 UCC Transmit-on-Demand Register (UTODR)

Refer to Section 22.3.4, “UCC Transmit-On-Demand Register (UTODR).”

23.3 UCC Buffer Descriptors (BDs)

Data associated with each UCC channel is stored in buffers and each buffer is referenced by a buffer descriptor (BD) that can reside anywhere in multi-user RAM. The total number of 8-byte BDs is limited only by the size of the multi-user RAM (128 BDs/1 Kbyte).

Each 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 QUICC Engine module 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 QUICC Engine module writes the length after the received data is placed into the associated buffer and the buffer is closed. In frame-based protocols (but not including UCC 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 QUICC Engine module 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.

The format of Tx and Rx BDs, shown in Figure 23-4, is the same in each UCC mode. Only the status and control bits differ for each protocol.

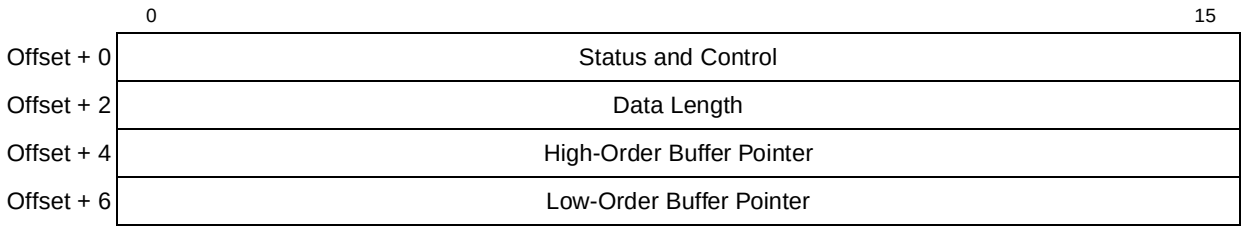


Figure 23-4. UCC Buffer Descriptors (BDs)

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 QUICC Engine module does not assume that all buffers of a single frame are currently linked to the BD table. The QUICC Engine module does assume, however, that the unlinked buffers are provided by the CPU 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 23-5 shows the UCC BD table and buffer structure.

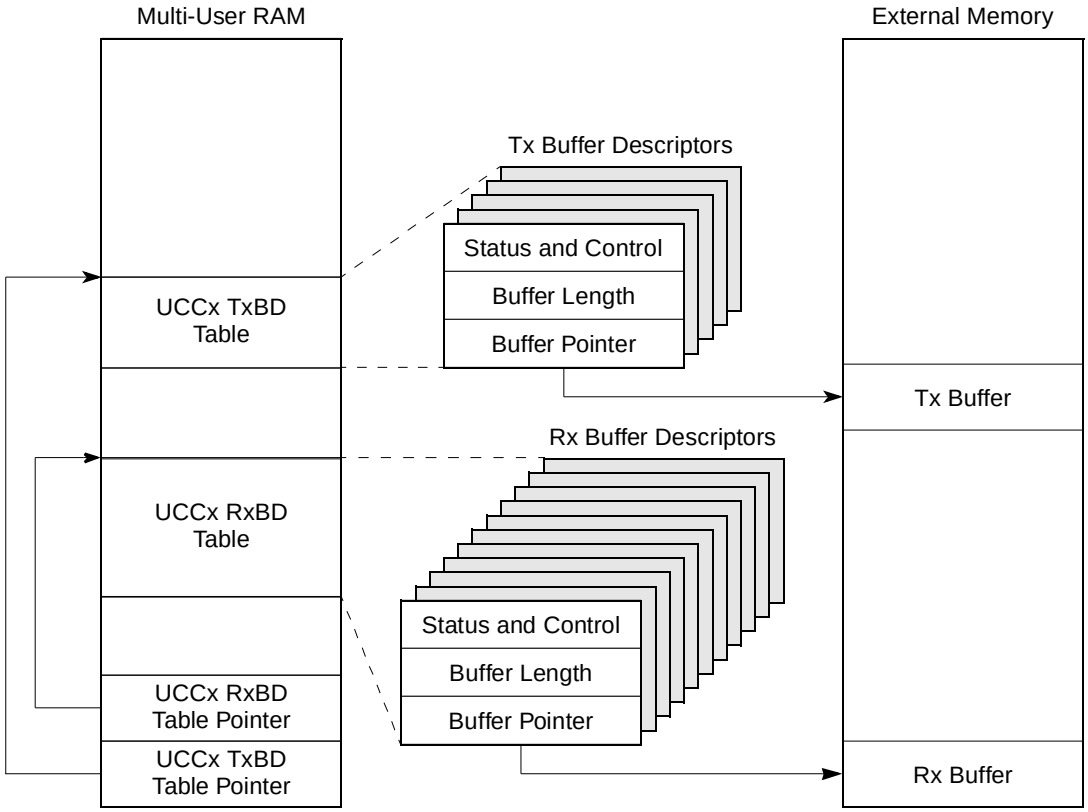


Figure 23-5. UCC BD and Buffer Memory Structure

In all protocols, BDs can point to buffers in the internal multi-user RAM. However, because multi-user RAM is used for descriptors, buffers are usually put in external RAM, especially if they are large.

The QUICC Engine module processes TxBDs straightforwardly; when the transmit side of a UCC is enabled, the QUICC Engine module starts with the first BD in that UCC TxBD table. Once the QUICC Engine module detects that the R bit is set in the TxBD, it starts processing the buffer. The QUICC Engine module detects that the BD is ready when it polls the R bit or when the user writes to the UTODR. After data from the BD is put in the Tx FIFO, if necessary the QUICC Engine module waits for the next descriptor's R bit to be set before proceeding. Thus, the QUICC Engine module does no look-ahead descriptor processing and does not skip BDs that are not ready. When the QUICC Engine module 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 QUICC Engine module 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 QUICC Engine module uses RxBDs similarly. When data arrives, the QUICC Engine module 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 QUICC Engine module does not move from the current BD until E is set by the core (the buffer is empty). After using a descriptor, the QUICC Engine module 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 QUICC Engine module 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.

23.4 UCC Parameter RAM

The UCC parameter RAM base address is defined in Section 22.2.1.1, "UCC Page Base Address." The protocol-specific part of the UCC parameter RAM is discussed in the specific protocol descriptions and the part that is common to all UCC protocols is shown in Table 23-4.

Some parameter RAM values must be initialized before the UCC can be enabled. Other values are initialized or written by the QUICC Engine module. 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.
- See Section 23.4.6, "Reconfiguring the UCC."

Table 23-4 shows the parameter RAM map for all UCC protocols. Boldfaced entries must be initialized by the user.

Table 23-4. UCC Parameter RAM Map for All Protocols

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address—offset from the beginning of multi-user RAM. The BD tables can be placed in any unused portion of the multi-user RAM. The QUICC Engine module 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 operation can occur if BD tables of active UCCs overlap. Values in RBASE and TBASE should be multiples of eight.
0x02	**TBASE**	Hword	
0x04	**RBMR**	Byte	Rx bus mode. See Section 23.4.1, "Bus Mode Registers (RBMR and TBMR)."
0x05	**TBMR**	Byte	Tx bus mode. See Section 23.4.1, "Bus Mode Registers (RBMR and TBMR)."
0x06	**MRBLR**	Hword	Maximum receive buffer length. Defines the maximum number of bytes the device writes to a receive buffer before it goes to the next buffer. The device 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 Ethernet and HDLC modes, and in totally transparent mode unless the Rx FIFO is 8-bits wide (GUMR_H[RFW] = 1). **Note:** Although MRBLR is not intended to be changed while the UCC 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 Rx BD 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]
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.

Table 23-4. UCC Parameter RAM Map for All Protocols (continued)

Offset [1]	Name	Width	Description
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 QUICC Engine module 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 QUICC Engine module 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.)

Note:

1 From UCC*x* base.

2 These parameters need not be accessed for normal operation but may be helpful for debugging.

3 For QUICC Engine module use only.

23.4.1 Bus Mode Registers (RBMR and TBMR)

There are separate bus mode registers for Rx buffers (RBMR) and for Tx buffers (TBMR). On the MPC826x devices, these were called function code registers, RFCR and TFCR. The bus mode registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 23-6 shows the register format. Boldfaced entries must be initialized by the user.

Address: UCC*x* base + 0x04 (RBMR*x*); UCC*x* base + 0x05 (TBMR*x*) Access: Read/Write

	0	1	2	3	4	5	6	7
R W	—		**GBL**	**BO**		**CETM**	**DTB**	**BDB**
Reset	All zeros							

Figure 23-6. Bus Mode Registers (RBMR and TBMR)

Table 23-5 describes RBMR*x*/TBMR*x* fields.

Table 23-5. RBMR*x* /TBMR*x* Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global. Indicates whether the memory operation should be snooped. 0 Snooping on the Coherent System Bus (CSB) is disabled. 1 Snooping on the Coherent System Bus (CSB) is enabled. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, "Serial DMA Mode Register (SDMR)."
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 and transparent) or at the beginning of the next BD. 00,01,11 Reserved modes. 10 Big-endian byte ordering. As data is sent onto the serial line from the data buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same buffer word. These bits must be set to 10 value.
5	**CETM**	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine module, for debug purposes. See Section 18.1.1, "Data Paths.
6	**DTB**	Indicates on what bus the data is located. 0 On the coherent system bus (CSB) 1 On the QUICC Engine secondary bus. The programming of this bit must be consistent with the System Configuration. See Figure 3-1 in System Interface chapter.
7	**BDB**	Indicates on what bus the BDs are located. 0 On the coherent system bus (CSB). 1 On the QUICC Engine secondary bus. The programming of this bit must be consistent with the System Configuration. See Figure 3-1.

23.4.2 UCC Event Register (UCCE)

Refer to Section 22.3.5, "UCC Event Register (UCCE)," and to the protocol specific chapters of the user manual.

23.4.3 UCC Mask Register (UCCM)

Refer to Section 22.3.6, "UCC Mask Register (UCCM)," and to the protocol specific chapters of the user manual.

23.4.4 UCC Status Register

.Refer to Section 22.3.7, "UCC Status Register," and to the protocol specific chapters of the user manual.

23.4.5 Initializing the UCCs

The UCCs 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 UCCs:

1. Write GUEMR register to UCC slow protocols (Value 0x0).
2. Write the parallel I/O ports to configure and connect the I/O pins to the UCCs.
3. Configure the parallel I/O registers to enable $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ if these signals are required.
4. If the time-slot assigner (TSA) is used, the serial interface (SI*x*) must be configured. If the UCC is used in NMSI mode, CMXUCR*x* has to be initialized.
5. Write all GUMR_L, GUMR_H bits but keep ENT or ENR cleared.
6. Write the PSMR.
7. Write the UDSR.
8. Initialize the required values for this UCC's parameter RAM.
9. Initialize the transmit/receive parameters via the QUICC Engine command register (CECR).
10. Clear out any current events in UCCE (optional).
11. Write ones to UCCM register to enable interrupts.
12. Set GUMR_L[ENT] and GUMR_L[ENR].

Descriptors can have their R or E bits set at any time. Notice that the CECR does not need to be accessed after a hardware reset. A UCC 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 CECR[RST].

23.4.6 Reconfiguring the UCC

The proper reconfiguration sequence must be followed for UCC parameters that cannot be changed dynamically. The steps in the following sections show how to disable, reconfigure and re-enable a UCC to ensure that buffers currently in use are properly closed before reconfiguring the UCC and that subsequent data goes to or from new buffers according to the new configuration.

Modifying parameter RAM does not require the UCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CECR[RST] to reset the entire QUICC Engine module.

23.4.6.1 General Reconfiguration Sequence for a UCC Transmitter

A UCC transmitter can be reconfigured by following these general steps:

1. If the UCC is sending data, issue a STOP TRANSMIT command. Transmission should stop smoothly. If the UCC 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 GUMR_L[ENT] to disable the UCC transmitter and put it in reset state.

3. Modify UCC 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 GUMR_L[ENT]. Transmission begins using the TxBD pointed to by TBPTR, assuming the R bit is set.

23.4.6.2 Reset Sequence for a UCC Transmitter

The following steps reinitialize a UCC transmit parameters to the reset state:

1. Clear GUMR_L[ENT].
2. Make any modifications then issue the INIT TX PARAMETERS command.
3. Set GUMR_L[ENT].

23.4.6.3 General Reconfiguration Sequence for a UCC Receiver

A UCC receiver can be reconfigured by following these steps:

1. Clear GUMR_L[ENR]. The UCC receiver is now disabled and put in a reset state. If a protocol switch is performed from fast protocol, follow step 2 in Section 23.4.6.5, “Switching Protocols.”
2. Modify UCC 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 GUMR_L[ENR]. Reception begins using the RxBD pointed to by RBPTR, assuming the E bit is set.

23.4.6.4 Reset Sequence for a UCC Receiver

To reinitialize the UCC receiver to the state it was in after reset, follow these steps:

1. Clear GUMR_L[ENR].
2. Make any modifications then issue the INIT RX PARAMETERS command.
3. Set GUMR_L[ENR].

23.4.6.5 Switching Protocols

To switch a UCC’s protocol without resetting the board or affecting other UCCs, follow these steps:

1. Clear GUMR_L[ENT, ENR].
2. If a protocol switch is preformed from fast protocol: Issue the Pushsched host command for UCC Transmitter (CECDR value 0x80). Issue the Pushsched host command for UCC Receiver (CECDR value 0x82).
3. Make protocol changes in the GUMR and additional parameters then issue the INIT TX and RX PARAMETERS command to initialize both Tx and Rx parameters.

4. Set GUMR_L[ENT, ENR] to enable the UCC with the new protocol.

23.4.7 Saving Power

To save power when not in use, a UCC can be disabled by clearing GUMR_L[ENT, ENR].

Chapter 24
UART Mode and Asynchronous HDLC

24.1 Introduction

NOTE

In MPC832x, the UART and Asynchronous HDLC protocols require a software patch from Freescale. Please contact an FAE.

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 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 24-1.

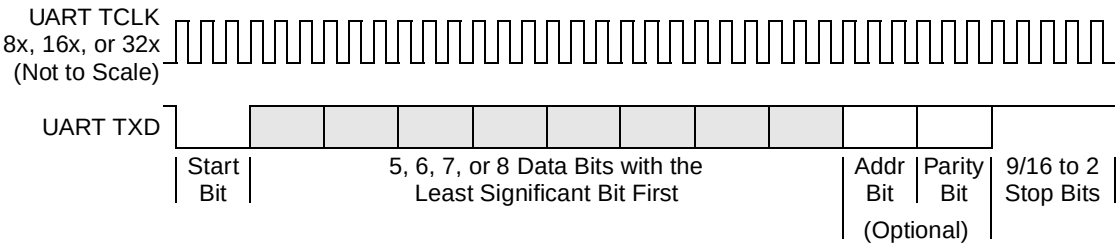


Figure 24-1. UART Character Format

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. The 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 the available UART protocols 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 a synchronous UART (isochronous operation), a separate clock signal is explicitly provided with the data. Start and stop bits are present in synchronous UARTs, but oversampling is not required because the clock is provided with each bit.

The general UCC mode register (GUMR) is used to configure a UCC 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, a UCC UART controller can communicate with any existing RS-232-type device and can provide a serial communication port to other microprocessors and terminals (either locally or via 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 wake-up 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.

24.1.1 Block Diagram

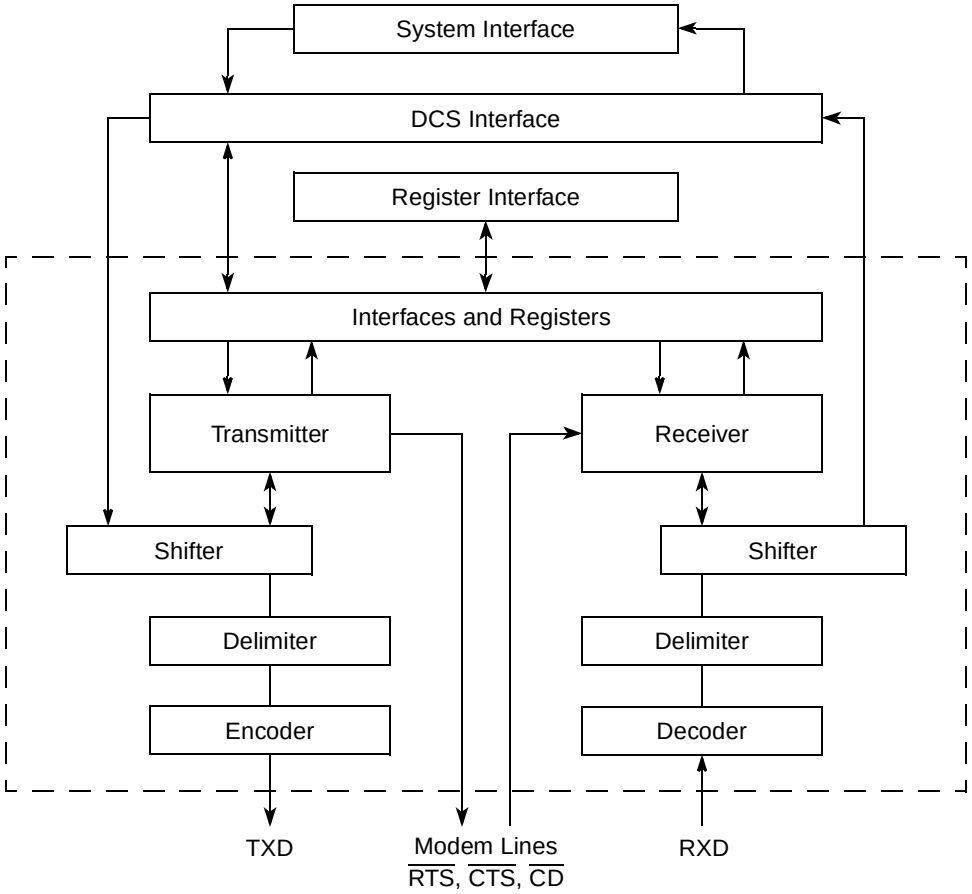


Figure 24-2. UART Block Diagram

24.1.2 Features

The following list summarizes the main features of a UCC UART controller:

- Flexible message-based data structure
- Implements synchronous and asynchronous UART
- Multidrop operation
- Receiver wake-up on idle line or address bit
- Receiver inserts 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

24.1.3 Modes of Operation

- Asynchronous mode
- Synchronous mode

24.1.3.1 Asynchronous Mode

In asynchronous mode (the default mode), the receive shift register receives incoming data on RXD*x*. Control bits in the UART mode register (UPSMR) 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 the bits do not agree, the noise indication counter (NOSEC) in the parameter RAM is incremented. When a complete character has been clocked in, the contents of the receive shift register are transferred to the receive DCS which is external to the UCC, before proceeding to the receive buffer. The QUICC Engine module flags UART events, including reception errors, in UCCE and the RxBD status and control fields.

The UCC can receive fractional stop bits. The next character's start bit can begin any time after the three middle samples are taken. For example, if GUMR_L[RDCR]=01 (x8 over sampling), the receiver is able to detect a start bit after the 5th sample of the stop bit (given that in this case the three middle samples are the 3rd,4th and 5th samples). 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 UCC. 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.

24.1.3.2 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 UPSMR 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 DCS before proceeding to the receive buffer. The QUICC Engine module flags UART events, including reception errors, in UCCE and the RxBD status and control fields.

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.

24.2 External Signal Descriptions

Table 24-1. Interface A—Detailed Signal Descriptions

Signal	I/O	Description	
TXD	O	Serial data out line	
		State Meaning	Asserted/Negated—for high/low value transmitted
		Timing	Assertion/Negation—according to serial clk TCLK for transmitter
$\overline{\text{RTS}}$	O	Transmitter is ready to transmit	
		State Meaning	Active low signal
		Timing	Assertion/Negation—according to serial clk RCLK for receiver
$\overline{\text{CTS}}$	I	Clear To send (transmitter is allowed to transmit)	
		State Meaning	Active low signal
		Timing	Assertion/Negation—according to serial clk TCLK for transmitter
RXD	I	Serial data In line	
		State Meaning	Asserted/Negated—for high/low value received
		Timing	Assertion/Negation—according to serial clk RCLK for receiver
$\overline{\text{CD}}$	I	Carrier Detect (data can be driven to receiver)	
		State Meaning	Active low signal
		Timing	Assertion/Negation—according to serial clk TCLK for transmitter

24.3 Memory Map/Register Definition

24.3.1 Overview

Table 24-2. Memory Map

Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x30[1]	UCC UART parameter RAM	R/W	0x0000_0000	24.3.2/24-6
As set in parameter RAM	UCC UART receive buffer descriptor (RxBD)	R/W	0x0000_0000	24.3.5/24-11
As set in parameter RAM	UCC UART transmit buffer descriptor (TxBD)	R/W	0x0000_0000	24.3.6/24-14
0xE[2]	UDSR data synchronization register (UDSR)	R/W	0x7E_7E	24.3.3/24-9
0x8[2]	UART mode register (UPSMR)	R/W	0x0000_0000	24.3.4/24-9
0x10[2]	UCC UART event register (UCCE)	R/W	0x0000_0000	24.3.7/24-15
0x14[2]	UCC UART mask register (UCCM)	R/W	0x0000_0000	24.3.7/24-15
0x17[2]	UCC UART status register (UCCS)	R/W	0x0000_0000	24.3.8/24-18

1 Offset from UCC's parameter RAM base address

2 Offset from UCC's register base address.

24.3.2 UCC UART Parameter RAM

For UART mode, the protocol-specific area of the UCC parameter RAM is mapped as shown in Table 24-3. Description of contents of the first 0x30 bytes can be found in Chapter 23, "UCC for Slow Protocols." The Offsets in this table are relative to the UCC_BASE address. The default base address (default page assignment) of the UCC is described in Section 19.2, "Parameter RAM."

Note: Bit fields in **boldface** must have their initial values set by the user.

Table 24-3. UART-Specific UCC 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.
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.

Table 24-3. UART-Specific UCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
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 QUICC Engine module. 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.
0x40	**FRMEC**	Hword	
0x42	**NOSEC**	Hword	
0x44	**BRKEC**	Hword	
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 = 0x002, 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	
0x4C	RTEMP	Hword	Temporary storage
0x4E	**TOSEQ**	Hword	Transmit out-of-sequence character. Inserts out-of-sequence characters, such as XOFF and XON, into the transmit stream. The TOSEQ character is put in the Tx DCS without affecting a Tx buffer in progress. See Section 24.5.4, “Inserting Control Characters into the Transmit Data Stream.”
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	
0x4C	RTEMP	Hword	Temporary storage
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.

Table 24-3. UART-Specific UCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x64	RLBC	Hword	Receive last break character. Used in synchronous UART when UPSMR[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 (exclusive) to where the first one was written. RLBC = 0b001xxxxxxxxxxxx indicates two zeros; 0b1xxxxxxxxxxxxxx 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).
0x66	—	Hword	Reserved
0x68	—	Word	Reserved. Should be cleared.
0x6C	—	Byte	Reserved. Should be cleared.
0x6D	—	3 bytes	Reserved. Should be cleared.
0x70	—	Word	Reserved. Should be cleared.
0x74	—	Word	Reserved. Should be cleared.
0x78	—	Word	Reserved. Should be cleared.
0x7C	—	Word	Reserved. Should be cleared.
0x80	—	Word	Reserved. Should be cleared.
0x84	—	Word	Reserved. Should be cleared.
0x88	—	Word	Reserved Should be cleared.
0x8C	—	Word	Reserved. Should be cleared.

Note:

1 From UCC_base. See Section 19.2, “Parameter RAM.”

24.3.3 UCC Data Synchronization Register (UDSR)

Address: UCC*x*_BASE + 0xE Access: Read/Write

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

Figure 24-3. UDSR Register

Table 24-4. UDSR Field Descriptions

Bits	Name	Description
0	—	Reserved
1–4	**FSB**	Fractional stop bits. For 16x 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 32x 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 8x 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. See description of GUMR.
5–15	—	Reserved. See default values after reset in Figure 24-3.

24.3.4 UART Mode Register (UPSMR)

For UART mode, the UCC protocol-specific mode register is called the UART mode register (UPSMR). Many bits can be modified while the receiver and transmitter are enabled. Figure 24-4 shows the UPSMR in UART mode.

Address: UCC*x*_BASE + 0x8 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FLC	SL	CL		UM		FRZ	RZS	SYN	DRT	—	PEN	RPM		TPM	
Reset	All zeros															

Figure 24-4. Protocol-Specific Mode Register for UART (UPSMR)

Table 24-5 describes PSMR UART fields.

Table 24-5. UPSMR UART Field Descriptions

Bits	Name	Description
0	FLC	Flow control. The GUMR_H[CTSS,CTSP]) registers determine the mode of $\overline{\text{CTS}}$. 0 Normal operation ($\overline{\text{CTS}}$ negation regarded as an error condition). 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 UCC sends. SL can be modified on-the-fly. The receiver is always enabled for one stop bit unless the UCC UART is in synchronous mode and UPSMR[RZS] is set. Fractional stop bits are configured in the DSR. 0 One stop bit 1 Two 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 QUICC Engine module 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 UCC completes transmission of the current character and then freezes. After FRZ is cleared, transmission resumes from the next character.
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 UCC 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.

Table 24-5. UPSMR UART Field Descriptions (continued)

Bits	Name	Description
8	SYN	Synchronous mode 0 Normal asynchronous operation. GUMR_L[TENC,RENC] must select NRZ and GUMR_L[TDCR, RDCR] select either 8x, 16x, or 32x. 16x is recommended for most applications. 1 Synchronous UCC UART controller using 1× clock (isochronous UART operation). GUMR_L[TENC, RENC] must select NRZ and GUMR_L[RDCR, TDCR] select 1x 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 UCC 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]. **Note:** If DRT = 1, GUMR_H[CDS] should be cleared unless both of the following are true: the same clock is used for TCLK and RCLK, and GUMR_H[CTS] either has synchronous timing or is always asserted.
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.

24.3.5 UCC UART Receive Buffer Descriptor (RxBD)

The QUICC Engine module uses RxBDs to report on each buffer received. The QUICC Engine module 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 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 24-5 shows an example of how RxBDs are used in receiving.

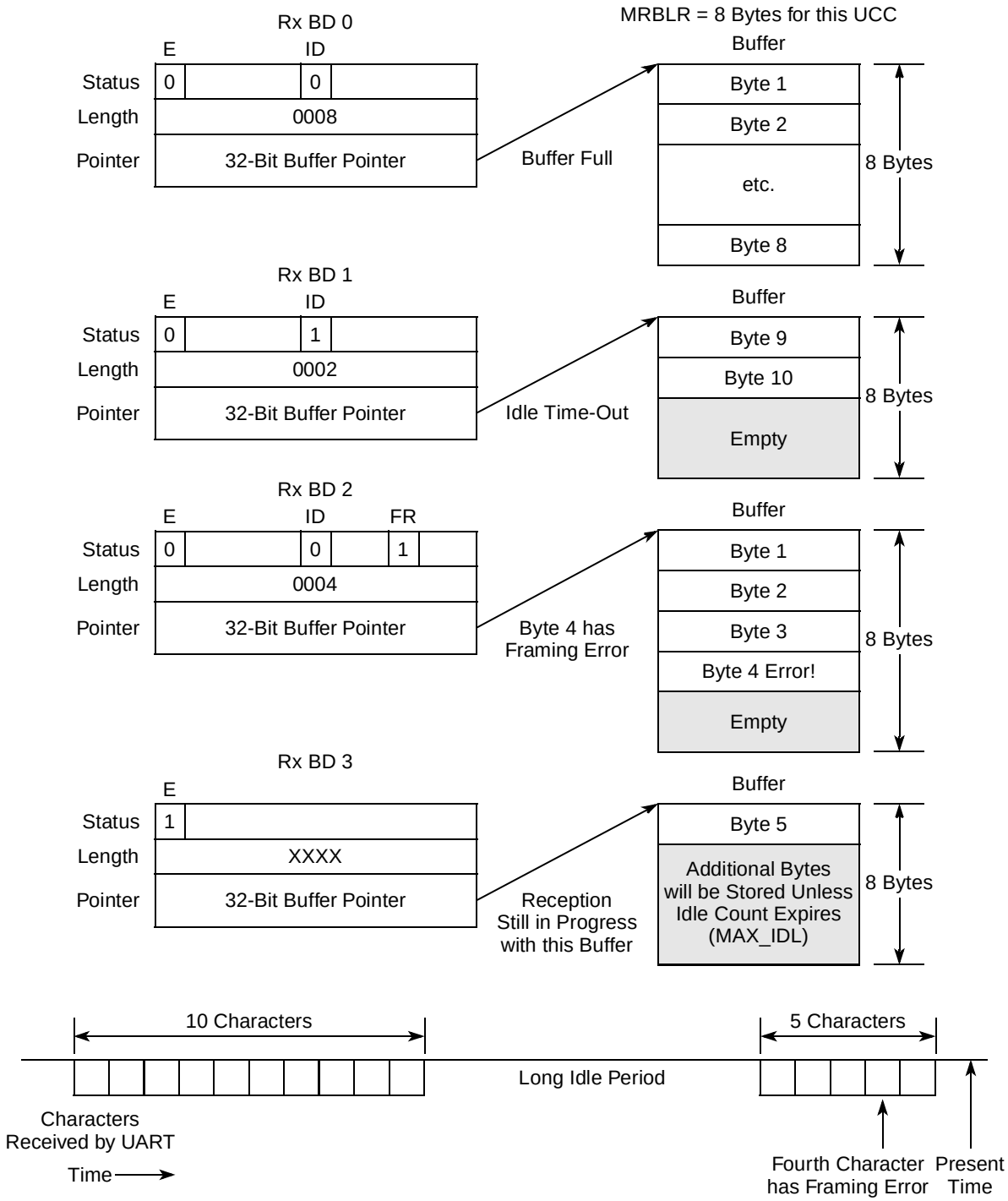


Figure 24-5. UCC UART Receiving using RxBDs

Figure 24-6 shows the UCC UART RxBD.

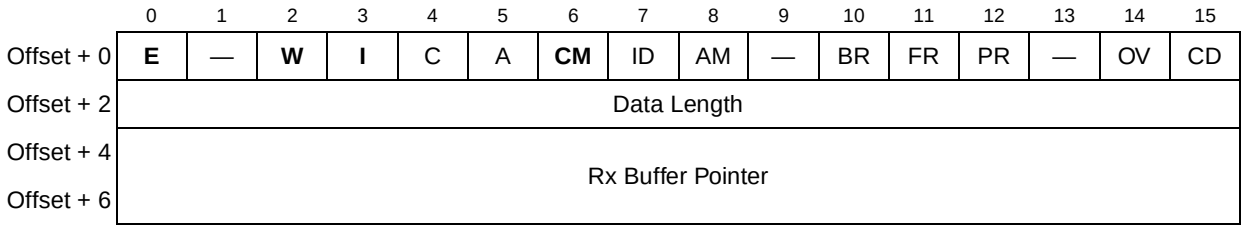


Figure 24-6. UCC UART Receive Buffer Descriptor (RxBD)

Table 24-6 describes RxBD status and control fields. Fields shown in bold type must have their initial values set by the user.

Table 24-6. UCC 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 QUICC Engine module does not reuse this BD while E = 0. 1 The buffer is not full. The QUICC Engine module 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 QUICC Engine module 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 multi-user RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is filled. 1 The RISC sets UCCE[RX] when this buffer is completely filled by the QUICC Engine module, indicating the need for the core to process the buffer. Setting UCCE[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 QUICC Engine module clears E after this BD is closed. 1 The QUICC Engine module does not clear E after this BD is closed, allowing the buffer to be overwritten when the QUICC Engine module 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. 0 Buffer was closed because MRBLR number of bytes have been received. 1 Buffer was closed because MAX_IDL idles were detected.

Table 24-6. UCC UART RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
8	AM	Address match. Significant only if the address bit is set and automatic multidrop mode is selected in UPSMR[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.
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 23.3, "UCC Buffer Descriptors (BDs)," describes the data length and buffer pointer fields.

24.3.6 UCC UART Transmit Buffer Descriptor (TxBD)

The QUICC Engine module uses BDs to confirm transmission and indicate error conditions so the CPU knows that buffers have been serviced. Figure 24-7 shows the UCC UART TxBD.

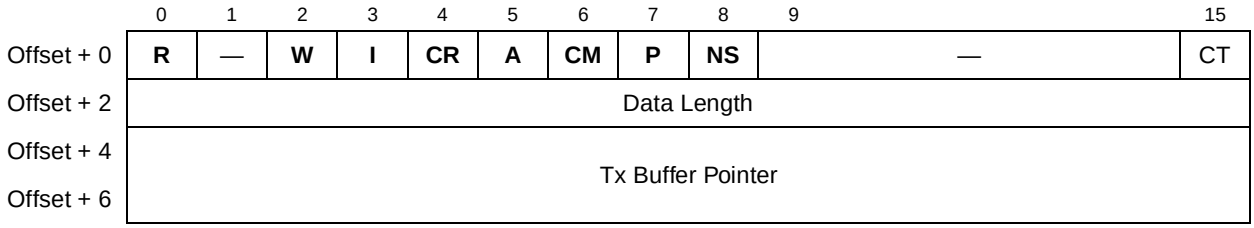


Figure 24-7. UCC UART Transmit Buffer Descriptor (TxBD)

Table 24-7 describes TxBD status and control fields. Fields shown in bold type must have their initial values set by the user.

Table 24-7. UCC 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 QUICC Engine module 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

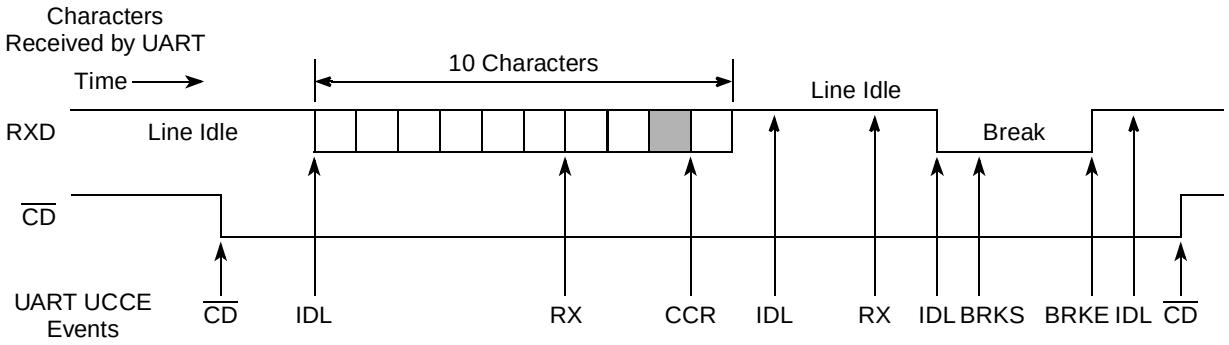
Table 24-7. UCC UART TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
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 Q UICC Engine module 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 multi-user RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is processed. 1 UCCE[TX] is set after this buffer is processed by the QUICC Engine module, which can cause an interrupt.
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 QUICC Engine module clears R after this BD is closed. 1 The QUICC Engine module does not clear R after this BD is closed, allowing the buffer to be resent next time the QUICC Engine module 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 UPSMR[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 24.5.7, “Fractional Stop Bits (Transmitter).”
9–14	—	Reserved, should be cleared
15	CT	$\overline{\text{CTS}}$ lost. The QUICC Engine module 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 23.3, “UCC Buffer Descriptors (BDs).”

24.3.7 UCC UART Event Register (UCCE) and Mask Register (UCCM)

UCCE is used to report events recognized by the UART channel and to generate interrupts. When an event is recognized, the controller sets the corresponding UCCE bit. Interrupts can be masked in UCCM, which has the same format as UCCE. Setting a mask bit enables the corresponding UCCE interrupt; clearing a bit masks it. Figure 24-8 shows example interrupts that can be generated by the UCC UART controller.

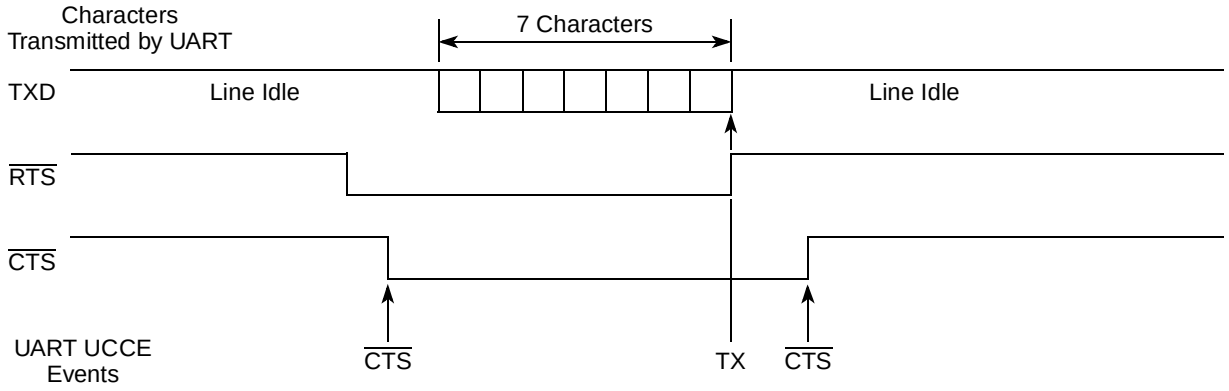


Notes:

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 'QUICC Engine ports interrupts' in the system interrupt controller not in the UCC itself.

Legend

A receive control character defined not to be stored in the Rx buffer.



Notes:

1. TX event assumes that all seven characters were put into a single buffer and TxBD[CR] = 1.
2. The $\overline{\text{CTS}}$ event must be programmed in the 'QUICC Engine ports interrupts' in the system interrupt controller not in the UCC itself.

Figure 24-8. UCC UART Interrupt Event Example

UCCE bits are cleared by writing ones; writing zeros has no effect. Unmasked bits must be cleared before the QUICC Engine module clears an internal interrupt request. Figure 24-9 shows UCCE/UCCM for UART operation.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM) Access: Read/Write

	0–5	6	7	8	9	10	11	12	13	14	15
R W	—	AB	IDL	GRA	BRKE	BRKS	—	CCR	BSY	TX	RX
Reset	All zeros										

Figure 24-9. UCC UART Event Register (UCCE) and Mask Register (UCCM)

Table 24-8 describes UCCE fields for UART mode.

Table 24-8. UCCE/UCCM Field Descriptions for UART Mode[1]

Bits	Name	Description
0–5	—	Reserved.[1]
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 20, "Multiplexing and Timers."
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 UCCS[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.[1]
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 DCS. 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.

Note:

1 Reserved bits in the UCCE should not be masked in the UCCM register.

24.3.8 UCC UART Status Register (UCCS)

UCCS, shown in Figure 24-10, monitors the real-time status of RXD.

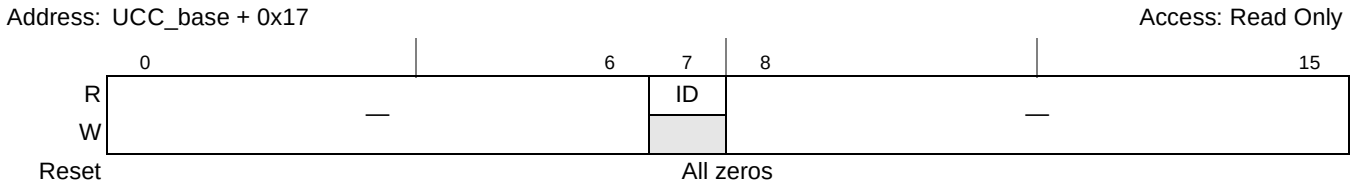


Figure 24-10. UCC Status Register for UART Mode (UCCS)

Table 24-9 describes UART UCCS fields.

Table 24-9. UART UCCS Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	ID	Idle status. Set when RXD has been a logical “one” for at least a full character time. 0 The line is not idle. 1 The line is idle.
8–15	—	Reserved, should be cleared.

24.4 Functional Description

24.4.1 Data-Handling Methods: Character- or Message-Based

A UCC UART controller uses the same BD table and buffer structures as other protocols used in the QUICC Engine module, and supports multi-buffer, message-based, single-buffer, and character-based operation.

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.

For character-based transfers, each character is sent with stop bits and parity and input into separate 1-byte buffers. A maskable interrupt is generated when each buffer is received.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 24.5.2, “Receiving Control Characters.”

24.4.2 Error and Status Reporting

Overrun, parity, noise, and framing errors are reported via 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.

24.5 UCC UART Commands

The transmit commands in Table 24-10 are issued to the QUICC Engine command register (CECR).

Table 24-10. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GUMR_L, the transmitter starts polling the first BD in the TxBD table every 8 Tx clocks. STOP TRANSMIT disables character transmission. If the UCC receives STOP TRANSMIT as a message is being sent, the message is aborted. The transmitter finishes sending data transferred to its DCS 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. UCCE[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 UPSMR, 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. Issued 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 24-11.

Table 24-11. 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 UCC is enabled in the GUMR, 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 DCS is not lost when this command is executed. This command will use the CECR[MCN]=0x04
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.

24.5.1 Multidrop Systems and Address Recognition

In multidrop systems, more than two stations can be on a network, each with a specific address. Figure 24-11 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 1 bit to distinguish an address character from standard data characters. Programmed in UPSMR[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.

The following figure depicts two modes of connecting multiple UARTS for multidrop.

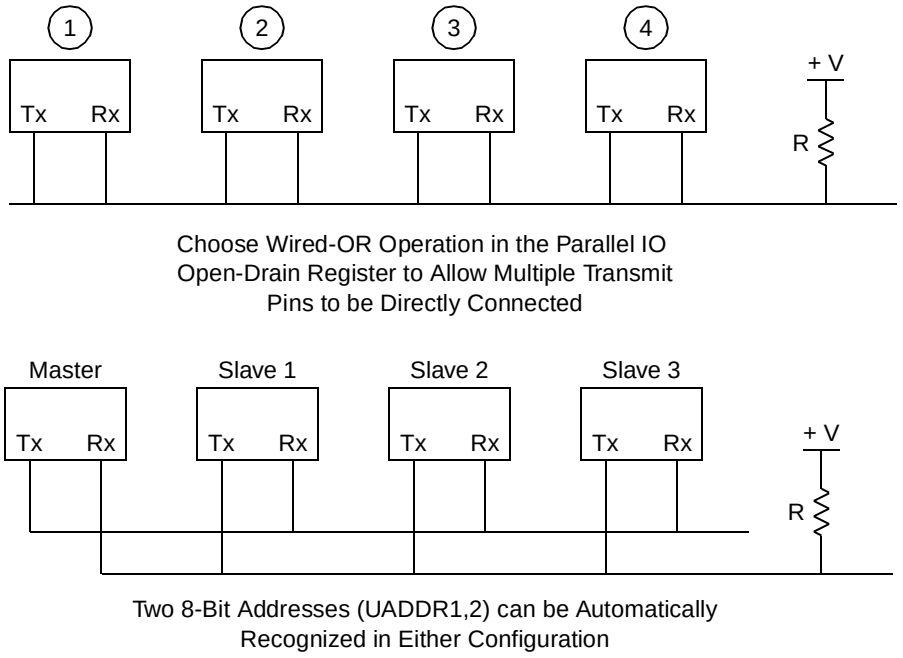


Figure 24-11. Two UART Multidrop Configurations

24.5.2 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 a 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 24-12.

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 UCC*x* base address

Figure 24-12. Control Character Table

Table 24-12 describes the data structure used in control character recognition. Items in bold must be initialized by the user.

Table 24-12. 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 UCCE[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 UCCE[CCR]. The current Rx buffer is not closed.
	2–7	—	Reserved
	8–15	**CHARACTER*n***	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.

24.5.3 Hunt Mode (Receiver)

A UART receiver in hunt mode remains deactivated until an idle or address character is recognized, depending on UPSMR[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 UPSMR[UM]. When a receiver in hunt mode receives a break sequence, it increments BRKEC and generates a BRK interrupt condition.

24.5.4 Inserting Control Characters into the Transmit Data Stream

The UCC 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 24-13, 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 (UCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

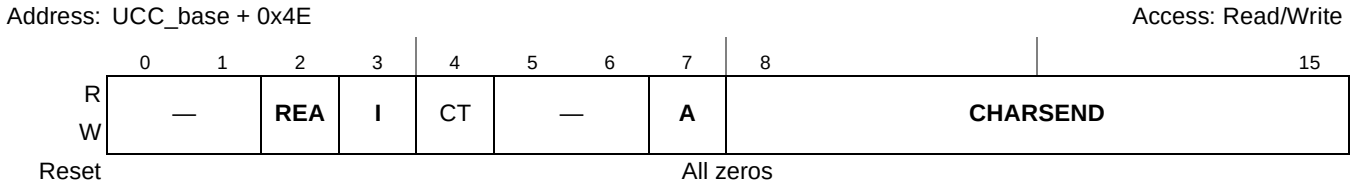


Figure 24-13. Transmit Out-of-Sequence Register (TOSEQ)

Table 24-13 describes TOSEQ fields.Fields shown in bold type should be initialized by the user.

Table 24-13. TOSEQ Field Descriptions

Bit	Name[1]	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 RISC clears this bit after transmission.
3	**I**	Interrupt. If this bit is set, transmission completion is flagged in the event register (UCCE[TX] is set), triggering a maskable interrupt to the core.
4	CT	Clear-to-send lost. Operates only if the UCC monitors $\overline{\text{CTS}}$ (GSMR_L[DIAG]). The RISC 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 24-13. TOSEQ Field Descriptions (continued)

Bit	Name[1]	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.

1 Fields shown in bold text must be initialized by the user.

24.5.5 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 UCC 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 DCS, they may not be sent for eight (UCC) or four (UCC) character times. To reduce this latency, set GUMR_H[TFL] to decrease the DCS size to one character before enabling the transmitter.

24.5.6 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 UCC sends a preamble sequence (idle character) before sending the buffer. When using the ctss mode (GUMR_H) the user should send at least one character of preamble in order not to lose the first two bits in the receiver. For example, for 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of ten 1’s is sent before the first character in the buffer.

24.5.7 Fractional Stop Bits (Transmitter)

The asynchronous UART transmitter 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.

24.5.8 Handling Errors in the UCC UART Controller

The UART controller reports character reception and transmission error conditions via the BDs, the error counters, and the UCCE. Modem interface lines can be monitored by the QUICC Engine module ports' interrupts pins. Transmission errors are described in Table 24-14.

Table 24-14. 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 RISC 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 UPSMR[FLC] in Section 24.3.4, "UART Mode Register (UPSMR)."

Reception errors are described in Table 24-15.

Table 24-15. Reception Errors

Error	Description
Overrun	Occurs when the channel overwrites the previous character in the Rx DCS 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	This error has the highest priority. 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. 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.

Table 24-15. Reception Errors (continued)

Error	Description
Framing	The UART reports a framing error 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 (UPSMR[RZS] = 1) when in synchronous mode (UPSMR[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 UCCE[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 UCCE[RX], which can generate an interrupt if not masked. If UPSMR[RZS] = 1 when the UART is in synchronous mode, a break sequence is detected after two successive break characters are received.

24.6 Asynchronous HDLC

Asynchronous HDLC uses HDLC framing techniques with UART-type characters. The asynchronous HDLC protocol is typically used as the physical layer for point-to-point protocol (PPP). Although asynchronous HDLC can be implemented in conjunction with the core, it is more efficient and less computationally intensive to let the RISC handle framing and transparency functions.

The RFC 1549 octet stuffing/unstuffing provided by this mode supports only asynchronous transmission. This mode cannot be used to provide octet stuffing for synchronous communication lines.

The following list summarizes the main features of the UCC in asynchronous HDLC mode:

- Flexible buffer structure lets all or part of a frame be sent or received
- Separate interrupts for received frames and transmitted buffers
- Automatic CRC generation and checking
- Support for nonmultiplexed serial interface control signals
- Automatic generation of opening and closing flags
- Reception of frames with a single shared flag
- Automatic generation and stripping of transparency characters according to RFC 1549 using transmit and receive control character maps
- Programmable Opening Flag, Closing Flag, and Control Escape characters
- Automatic transmission of the abort sequence after a STOP TRANSMIT command
- Automatic transmission of idle characters between frames and between characters

24.7 Asynchronous HDLC Frame Transmission Processing

The UCC in asynchronous HDLC mode (asynchronous HDLC controller) works with minimal core intervention. When the core enables the transmitter and sets TxBD[R] in the first BD of the table, the asynchronous HDLC controller fetches data from memory and starts sending the frame. If the current TxBD[L] is set (last buffer of a frame), the CRC and closing flag are appended. If TxBD[CM] is zero, the

transmitter updates frame status bits in the BD and clears TxBD[R]. If TxBD[I] is set, the controller sets UCCE[TXB] so an interrupt can be generated after each buffer, after a group of buffers, or after each frame is sent.

If TxBD[CM] is set, the asynchronous HDLC transmitter updates frame status bits in the BD after transmission but does not clear TxBD[R]. The transmitter then proceeds to the next TxBD and if necessary waits until it is ready. As the transmitter sends data, it performs the transparency encoding specified by the protocol. See Section 24.9, "Transmitter Transparency Encoding."

Figure 24-14. Asynchronous HDLC Frame Structure

BOF	Address	Control	Information	FCS (CRC)	EOF
8 bits	8 bits	8 bits	M * 8 bits	2 • 8 bits	8 bits

To rearrange buffers, such as for error handling or to expedite data ahead of previously linked buffers, issue a STOP TRANSMIT command before modifying the TxBD table or directly changing the current TxBD pointer TBPTR. When the asynchronous HDLC controller receives a STOP TRANSMIT command, it stops the transmission and sends the asynchronous HDLC abort sequence. It then sends idle characters until the RESTART TRANSMIT command is given, at which point it resumes transmission with the next TxBD.

24.8 Asynchronous HDLC Frame Reception Processing

The asynchronous HDLC receiver is designed to work with minimal core intervention. It can decode transparency characters, check the CRC of the frame, and detect errors on the line and in the controller. When the core enables the receiver and the receiver detects a data byte of the incoming frame preceded by one or more opening flags, the asynchronous HDLC controller fetches the next BD. If RxBD[E] is set, the controller starts transferring the incoming frame into the buffer. When the buffer is full, the controller clears RxBD[E]. If the incoming frame is larger than the buffer, the controller fetches the next BD, and if E is set, continues transferring the rest of the frame into its buffer.

The receiver decodes the transparency character required by asynchronous HDLC protocol as described in Section 24.10, "Receiver Transparency Decoding." When the frame ends, the controller checks the incoming CRC field and writes it to the buffer. The controller then updates RxBD[Data Length] with the total frame length, including the CRC bytes. The controller sets RxBD[L], writes the frame status bits, and clears RxBD[E] (if RxBD[CM] is zero). It then sets UCCE[RXF], which indicates that a frame was received and is in memory. The controller then waits for the start of the next frame, which may or may not have an opening flag.

24.9 Transmitter Transparency Encoding

The asynchronous HDLC transmitter encodes characters according to RFC 1549, a de facto standard of the Internet Engineering Task Force (IETF). It examines outgoing bytes and performs the transparency algorithm for the following conditions:

- The byte is a flag (0x7E for PPP)
- The byte is a control-escape character (0x7D)

- The byte value is between 0x00 and 0x1F and the corresponding bit in the Tx control character table is set

When a condition applies, a two-byte sequence is sent instead of the byte. The sequence consists of the control-escape character (0x7D) followed by the original byte exclusive-ORed with 0x20.

24.10 Receiver Transparency Decoding

The asynchronous HDLC receiver decodes characters according to RFC 1549. To recover the original data, it examines incoming data bytes and performs the transparency algorithm in the following ways:

- It discards characters whose corresponding bit is set in the Rx control character map. This character is assumed to have been inserted in the character stream by an intermediate device and is not part of the original frame.
- It reverses the transmission transparency sequence by discarding a received control-escape character (0x7D) and exclusive-ORing the following byte with 0x20 before performing the CRC calculation and writing the byte into memory.

Figure 24-15 shows the algorithm because some cases are not covered by RFC 1549.

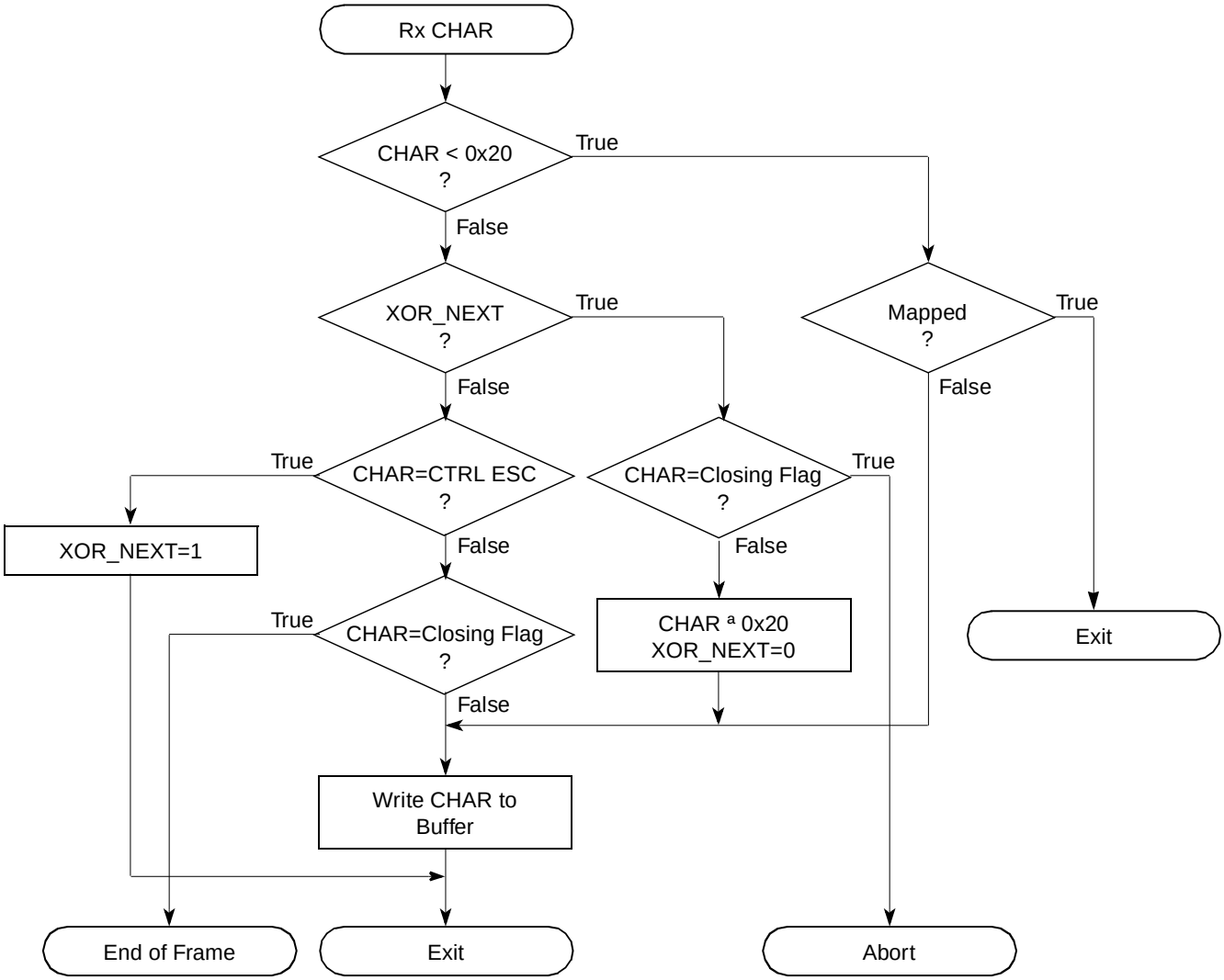


Figure 24-15. Receive Flowchart

24.11 Exceptions to RFC 1549

- An unmapped control character that follows 0x7D is modified by the XOR process. The CRC check should catch this.
- In addition to the abort sequence, frames are terminated by the following errors:
  - $\overline{\text{CD}}$ (carrier detect) lost
  - Receiver overrun
  - Framing error
  - Break sequence
- If an invalid sequence(0x7D7D) is received, the first control escape character is discarded, and the second is unconditionally XORed with 0x20. The sequence is thus stored in the buffer as 0x5D.

24.12 Asynchronous HDLC Channel Implementation

The following points are specific to asynchronous HDLC channel implementation:

- Flag sequence—The transmitter automatically generates the opening and closing flags. The receiver removes opening and closing flags before writing a frame to memory and receives frames with only one shared flag between frames, ignoring multiple flags.
- Address field—Neither generated nor examined by the microcode while sending or receiving. The destination address field of the frame must be included in the Tx buffer. Any address field compression, expansion, or checking must be performed by the core.
- Control field—Neither generated nor examined by the microcode during a transfer. The control field of the frame must be included in the buffer. Any control field compression, expansion, or checking is done by the core.
- Frame check sequence (FCS)—When sending, the FCS is appended to the frame before the closing flag is sent. The FCS is generated on the original frame before transparency characters, start/stop bits, or flags are added. When receiving, the FCS is checked automatically and calculated after any transparency characters, start/stop bits, and flags are removed. For both, the controller uses only a 16-bit CRC-CCITT polynomial.
- Encoding—The asynchronous HDLC controller supports 8 data bits, one start bit, one stop bit, and no parity. Program UPSMR[CHLN] to 0b11 for proper operation.
- Idle characters—When sending, the asynchronous HDLC controller sends idle characters when no data is available; when receiving, it ignores idle characters.

24.13 Asynchronous HDLC Mode Parameter RAM

For asynchronous HDLC mode, the protocol-specific area of the UCC parameter RAM is mapped as in Table 24-16.

Table 24-16. Asynchronous HDLC-Specific UCC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**C_MASK**	Word	CRC constant. Initialize with 0x0000_F0B8.
0x38	**C_PRES**	Word	CRC preset. Initialize with 0x0000_FFFF.
0x3C	**BOF**	Hword	Beginning-of-flag-character. Initialize to PPP-0x7E.
0x3E	**EOF**	Hword	End-of-flag character. Initialize to PPP-0x7E.
0x40	**ESC**	Hword	Control escape character. Initialize to 0x7D for PPP.
0x42	—	Word	Reserved
0x46	**ZERO**	Hword	Clear this field.
0x48	—	Hword	Reserved
0x4A	**RFTHR**	Hword	Received frames threshold. Number of Rx frames needed to trigger UCCE[RXF]
0x4C	—	Word	Reserved

Table 24-16. Asynchronous HDLC-Specific UCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x50	**TXCTL_TBL**	Word	Control character tables. Stores the bit array used for the Tx/Rx control characters. See Figure 24-16. Each bit corresponds to a character that should be mapped according to RFC 1549. If a TXCTL_TBL bit is set, its corresponding character is mapped; otherwise, it is not mapped. If an RXCTL_TBL bit is set, its corresponding character is discarded if received; otherwise, it is received normally.
0x54	**RXCTL_TBL**	Word	
0x58	**NOF**	Hword	Number of opening flags to be sent at the beginning of a frame. A value of n corresponds to n+1 flags.
0x5A – 0x100	—	—	Reserved

1 From UCC base.

Figure 24-16 shows bit arrangements for TXCTL_TBL and RXCTL_TBL.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x1F	0x1E	0x1D	0x1C	0x1B	0x1A	0x19	0x18	0x17	0x16	0x15	0x14	0x13	0x12	0x11	0x10

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0x0F	0x0E	0x0D	0x0C	0x0B	0x0A	0x09	0x08	0x07	0x06	0x05	0x04	0x03	0x02	0x01	0x00

Figure 24-16. TXCTL_TBL/RXCTL_TBL

24.14 Configuring GUMR and UDSR for Asynchronous HDLC

General UCC parameters can be configured as described in Chapter 23, "UCC for Slow Protocols" except for the following changes to the general UCC mode register and the UCC data synchronization register.

24.14.1 General UCC Mode Register (GUMR)

Table 24-17 shows asynchronous HDLC-specific information for the GUMR.

Table 24-17. Asynchronous HDLC-Specific GUMR Field Descriptions

Name	Description
RFW	Rx FIFO width (GUMR_H[26]) 0 Do not use. 1 Low-latency operation—for character-oriented protocols like UART, BISYNC, and asynchronous HDLC. The Tx FIFO is 8 bits wide and the Rx FIFO is one-fourth its normal size. This allows each character to be written to the buffer without waiting for 32 bits to be received.
TDCR/ RDCR	Tx/Rx divide clock rate (GUMR_L[14–15/16–17]). For asynchronous HDLC mode, 8×, 16×, or 32× must be chosen. Set TDCR = RDCR in most applications. 00 Do not use. 01 8× clock mode. 10 16× clock mode. 11 32× clock mode.

24.14.2 UCC Data Synchronization Register (UDSR)

The UCC data synchronization register (UDSR) is reserved in asynchronous HDLC mode. It should be left in its reset state of 0x7E7E.

24.15 Programming the Asynchronous HDLC Controller

Asynchronous HDLC mode is selected for a UCC by writing GUMR_L[MODE] = 0b0110. The asynchronous HDLC controller uses the same buffer and BD data structure as other modes and supports multibuffer operation. Receive errors are reported through the RxBD; transmit errors are reported through the TxBD. Status line information ($\overline{\text{CD}}$ and $\overline{\text{CTS}}$) is reported through the parallel I/O pins; a maskable interrupt is generated when the status of either line changes.

24.16 Asynchronous HDLC Commands

The transmit and receive commands are issued to the RISC command register (CECR).

Transmit commands are described in Table 24-18. After a hardware or software reset and a channel is enabled in the GUMR, the transmitter starts polling the first BD in the TxBD table every 8 transmit clocks, or immediately if TODR[TOD] = 1, and begins sending data if TxBD[R] is set.

Table 24-18. Transmit Commands

Command	Description
STOP TRANSMIT	Sends the asynchronous HDLC abort sequence (0x7D;0x7E for PPP, 0x7D) and disables data transmission. If the asynchronous HDLC controller receives this command during frame transmission, the abort sequence is put in the FIFO and the transmitter does not try to send more data from the current BD or advance to the next TxBD. The BD to be terminated is indicated by the TBPTR entry in the parameter RAM table. Note that unlike with other UCC protocols, the STOP TRANSMIT command does not flush the FIFO. Up to 32 characters can be sent ahead of the abort sequence unless GUMR_H[TFL] = 1.
GRACEFUL STOP TRANSMIT	Not supported by the asynchronous HDLC controller.
RESTART TRANSMIT	Re-enables transmission of characters; the asynchronous HDLC controller expects it after a STOP TRANSMIT command or transmitter error. The controller continues sending from the first character in the buffer using the current TxBD (pointed to by TBPTR).
INIT TX PARAMETERS	Initializes all Tx parameters in this channel's parameter RAM to reset state. It must be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command resets both Tx and Rx parameters.

Table 24-19 describes receive commands. After a hardware or software reset, and a channel is enabled in the GUMR, reception begins with the first BD in the RxBD table.

Table 24-19. Receive Commands

Command	Description
ENTER HUNT MODE	Forces the asynchronous HDLC controller to close the current RxBD, if it is in use, and enter hunt mode. Reception resumes after the controller finds a frame preceded by one or more opening flags.
CLOSE RXBD	Not supported by the asynchronous HDLC controller.
INIT RX PARAMETERS	Initializes all Rx parameters in the channel's parameter RAM to reset state. Issue only when the receiver is disabled. The INIT TX AND RX PARAMETERS command resets both Tx and Rx parameters.

24.17 Handling Errors in the Asynchronous HDLC Controller

The asynchronous HDLC controller reports frame reception and transmission error conditions using the channel BDs and the asynchronous HDLC event register (UCCE). Table 24-20 describes transmit errors.

Table 24-20. Transmit Errors

Error	Description
$\overline{\text{CTS}}$ Lost during Frame Transmission	The channel stops sending the buffer, closes it, sets UCCE[TXE] and TxBD[CT]. The channel resumes sending from the next TxBD after a RESTART TRANSMIT command is issued.

Table 24-21 describes reception errors.

Table 24-21. Receive Errors

Error	Description
Overrun	Overrun occurs when the RISC cannot keep up with the data rate or the SDMA channel cannot write the received data to memory. The previous data byte and frame status are lost. The controller closes the buffer and sets RxBD[OV] and UCCE[RXF]. The receiver then looks for the next frame.
$\overline{\text{CD}}$ Lost during Frame Reception	The channel stops receiving frames, closes the buffer, and sets UCCE[RXF] and RxBD[CD]. This error has highest priority. The rest of the frame is lost and other errors are not checked in that frame. The receiver then searches for the next frame once $\overline{\text{CD}}$ is reasserted.
Abort Sequence	When an abort sequence (0x7D, 0x7E for PPP) is detected, the channel closes the buffer by setting UCCE[RXF] and RxBD[AB]. CRC error status is not checked on aborted frames. If no frame is being received, the next BD is opened and then closed with RxBD[AB] set.
CRC	The channel writes the received cyclic redundancy check to the buffer, closes the buffer, and sets UCCE[RXF] and RxBD[CR]. After receiving this error, the receiver prepares to receive the next frame.
Break Sequence Received	The receiver detected the first character in a break sequence. The channel closes the buffer and sets UCCE[RXF] and RxBD[BRK]. CRC error status is not checked. UCCE[BRKS] is set when the first break of a sequence is found; UCCE[BRKE] is set when an idle bit is received after a break sequence.

24.18 UCC Asynchronous HDLC Registers

The following sections describe the UCC registers when in asynchronous HDLC mode.

24.18.1 Asynchronous HDLC Event Register (UCCE)/ Asynchronous HDLC Mask Register (UCCM)

The UCC event register (UCCE) is used as the asynchronous HDLC event register to generate interrupts and report events recognized by the asynchronous HDLC channel. When an event is recognized, the asynchronous HDLC controller sets the corresponding UCCE bit. Interrupts can be masked by clearing the appropriate bit in the asynchronous HDLC mask register (UCCM). UCCE bits, shown in Figure 24-17, are cleared by writing ones—writing zeros has no effect. Unmasked UCCE bits must be cleared before the QUICC Engine module clears the internal interrupt request.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM) Access: Read/Write

	0–2	3	4	5–6	7	8	9	10	11	12	13	14	15
R	—	GLR	GLT	—	IDL	—	BRKE	BRKS	TXE	RXF	BSY	TXB	RXB
W													
Reset	All zeros												

Figure 24-17. Asynchronous HDLC Event Register (UCCE)/Asynchronous HDLC Mask Register (UCCM)

Table 24-22 describes UCCE/UCCM fields.

Table 24-22. UCCE/UCCM Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3	GLR	Glitch on Rx. Set when the UCC finds a Rx clock glitch.
4	GLT	Glitch on Tx. Set when the UCC finds a Tx clock glitch.
5–6	—	Reserved, should be cleared.
7	IDL	Idle sequence status changed. Set when serial line status changes. Real-time status can be read in UCCS[ID].
8	—	Reserved, should be cleared.
9	BRKE	Break end. Marks the end of a break sequence—set when an idle bit is detected after a break sequence.
10	BRKS	Break start. Set when the first break character of a break sequence is received. Only one BRKS event occurs per break sequence, no matter the length of the sequence.
11	TXE	Tx error. Set when an error occurs on the transmitter channel.
12	RXF	Rx frame. Set when the number of frames specified in RFTHR are received. RXF is set no sooner than when the midpoint of the closing flag's stop bit arrives.
13	BSY	Busy condition. Set when a frame is received and discarded due to a buffer shortage.

Table 24-22. UCCE/UCCM Field Descriptions (continued)

Bits	Name	Description
14	TXB	Transmit buffer. Set when a buffer with TxBD[I] set is sent on the channel, not before 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 Tx FIFO.
15	RXB	Rx buffer. Set when a buffer with RxBD[I] set and RxBD[L] cleared is received over the channel.

24.18.2 UCC Asynchronous HDLC Status Register (UCCS)

The UCC asynchronous HDLC status register (UCCS), shown in Figure 24-18, monitors the real-time status of RXD. The real-time status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ is part of the parallel I/O.

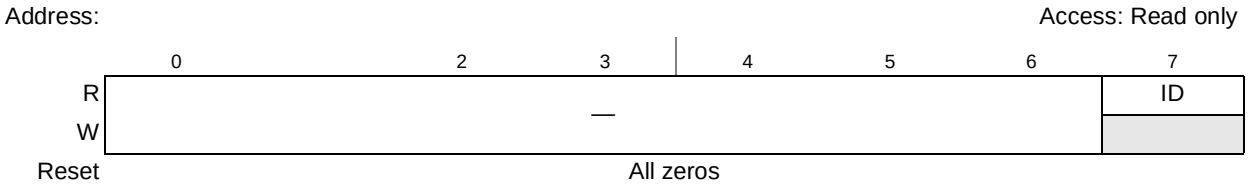


Figure 24-18. UCC Status Register for Asynchronous HDLC Mode (UCCS)

Table 24-9 describes asynchronous HDLC UCCS fields.

Table 24-23. Asynchronous HDLC UCCS 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.

24.18.3 Asynchronous HDLC Mode Register (UPSMR)

When the UCC is in asynchronous HDLC mode, the UPSMR, shown in Figure 24-19, acts as the asynchronous HDLC mode register.

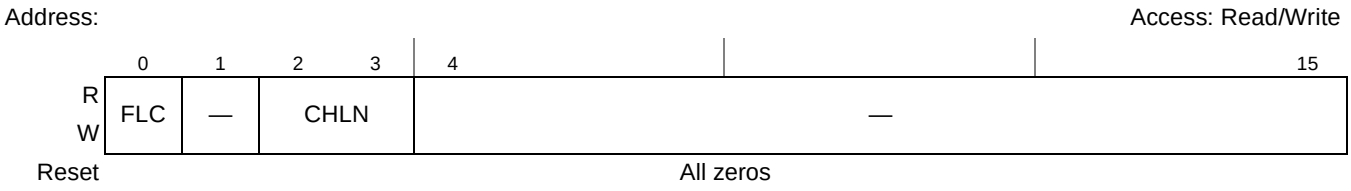


Figure 24-19. Asynchronous HDLC Mode Register (UPSMR)

Table 24-24 describes UPSMR fields.

Table 24-24. UPSMR Field Descriptions

Bits	Name	Description
0	FLC	Flow control 0 Normal operation ($\overline{\text{CTS}}$ negation regarded as an error condition). 1 Asynchronous flow control. When $\overline{\text{CTS}}$ is negated, the transmitter stops at the end of the current character. If $\overline{\text{CTS}}$ remains negated past the middle of the character, the next full character is sent before transmission stops. If $\overline{\text{CTS}}$ is reasserted, transmission resumes from where it stopped and no $\overline{\text{CTS}}$ lost error is reported. Only idle characters are sent while $\overline{\text{CTS}}$ is negated.
1	—	Reserved, should be cleared.
2–3	CHLN	Character length. On other protocols CHLN is the number of data bits in a character. For asynchronous HDLC mode, CHLN must be set to 0b11 (indicating a character length of 8 bits).
4–15	—	Reserved, should be cleared.

24.19 UCC Asynchronous HDLC RxBDs

The QUICC Engine module uses the RxBD, shown in Figure 24-20, to report on received data.

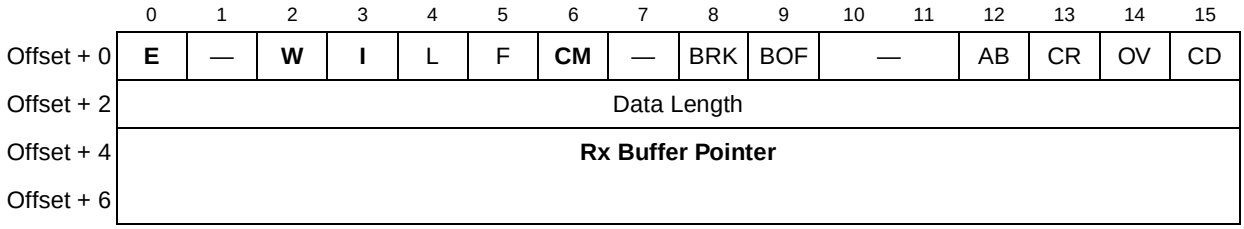


Figure 24-20. UCC Asynchronous HDLC RxBDs

Table 24-25 describes the UCC asynchronous HDLC RxBD status and control fields.

Table 24-25. Asynchronous HDLC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or stops receiving because of an error. The core can read or update any fields of this RxBD. The QUICC Engine module cannot reuse this BD while E = 0. 1 The buffer is not full. The RISC controls the BD and buffer. The core should not update the BD.
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 QUICC Engine module receives incoming data using the BD pointed to by RBASE. The number of RxBDs in a table is determined by the W bit.
3	**I**	Interrupt. 0 UCCE[RXB] is not set after this buffer is used. UCCE[RXF] is unaffected. 1 UCCE[RXB] or UCCE[RXF] is set when this buffer is used by the asynchronous HDLC controller.

Table 24-25. Asynchronous HDLC RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
4	L	Last in frame. 0 Not the last buffer in a frame. 1 Set by UCC when a buffer is the last in a frame which happens when a closing flag or error is received. If an error occurs, one or more of the BRK, CD, OV, BOF, CR, and AB bits are set. The UCC updates RxBD[Data Length].
5	F	First in frame. Set by the UCC 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 RISC does not clear E after the BD is closed allowing a buffer to be overwritten when the RISC next accesses the BD. However, E is cleared if an error other than CRC occurs during reception, regardless of CM.
7	—	Reserved, should be cleared.
8	BRK	Break character received. Set when a frame is closed because a break character is received.
9	BOF	Beginning of frame. Set when a frame is closed because a BOF character is received instead of the expected EOF.
10–11	—	Reserved, should be cleared.
12	AB	Rx abort sequence. Set when an abort sequence or framing error terminates a frame.
13	CR	Rx CRC error. Set when a frame has a CRC error. Received CRC bytes are written to the buffer.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost. Set when $\overline{CD}$ is negated during frame reception.

Because asynchronous 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.

24.20 UCC Asynchronous HDLC TxBDs

The QUICC Engine module uses the TxBD, shown in Figure 24-21, to confirm transmissions and indicate error conditions.

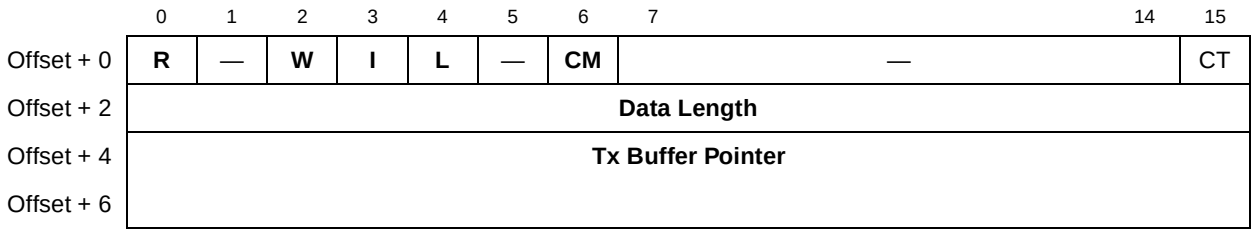


Figure 24-21. UCC Asynchronous HDLC TxBDs

Table 24-26 describes the UCC asynchronous HDLC TxBD status and control fields.

Table 24-26. Asynchronous HDLC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission; the BD and the buffer can be updated. The QUICC Engine module clears R after the buffer is sent or after an error condition. 1 The buffer is ready but is not sent or is being sent. Do not update the BD while R =1.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in table). 0 Not the last BD in the table. 1 The last BD in the table. After this buffer is used, the QUICC Engine module sends incoming data using the BD pointed to by TBASE. The number of TxBDs in this table are determined only by the W bit.
3	**I**	Interrupt. 0 UCCE[TXB] is not set after this buffer is sent. 1 UCCE[TXB] is set when this buffer is sent by the asynchronous HDLC controller.
4	**L**	Last. 0 Not the last buffer in the current frame. 1 Last buffer in the current frame. The proper CRC and closing flag are sent after the last byte.
5	—	Reserved, should be cleared.
6	**CM**	Continuous mode. 0 Normal operation. 1 The RISC does not clear R after this BD is closed, allowing its buffer to be resent when the RISC next accesses this BD. However, R is cleared if an error occurs during transmission, regardless of CM.
7–14	—	Reserved, should be cleared.
15	CT	$\overline{\text{CTS}}$ lost. In NMSI mode, $\overline{\text{CTS}}$ is lost during frame transmission. If more than one buffer has data in the FIFO when this error occurs, CT is set in the currently open TxBD. Written by the asynchronous HDLC controller after it finishes sending the buffer.

24.21 Differences Between HDLC and Asynchronous HDLC

The basic differences between HDLC and asynchronous HDLC modes are as follows:

- Asynchronous HDLC does not support the GRACEFUL STOP TRANSMIT command.
- Because asynchronous HDLC has no maximum received frame length counter, it receives all characters between opening and closing flags. There is no way to keep it from writing to memory. This does not affect the number of bytes received into a specific BD. A frame over the maximum length is received into memory in its entirety.
- If an error causes a frame to stop being received, the character being received at the moment the error occurred is not written into memory. For example, if a $\overline{\text{CD}}$ lost error occurs, the frame is closed and the partial character is not written to memory. Thus, the octet count reflects only the number of bytes written to memory.
- The automatic error counters in the HDLC controller are not implemented in the asynchronous HDLC controller.
- Noisy characters (characters for which all three samples are not identical) are not accounted for in the asynchronous HDLC controller. It is assumed that the CRC catches any data integrity problems.

24.22 Asynchronous HDLC Multi-User RAM Usage

The multiuser RAM consumption is dependent on several parameters. This section presents the way the multiuser RAM usage can be calculated according to the setting of these parameters.

Table 24-27 and Table 24-28 present the various parameters that impact the multi-user RAM usage.

Table 24-27. Tx and Rx Parameter RAM Usage

Data Structure	Size [Bytes]	Multiply by
Global Parameter RAM	256	1

Table 24-28. Internal Buffer Descriptors

Data Structure	Size [Bytes]	Multiply by
Tx buffer descriptors	8	Number of Tx Buffer Descriptors
Rx buffer descriptors	8	Number of Rx Buffer Descriptors

Table 24-29 presents the overall usage of multi-user RAM.

Table 24-29. Asynchronous HDLC Multi-user RAM Usage

Data Structure	Example Size [Bytes]
Parameter RAM usage	256
Tx buffer descriptors	8 × number of Tx BD's
Rx buffer descriptors	8 × number of Rx BD's
Total MRAM usage	256 + 8 × (Tx + Rx BD's)

Chapter 25
BISYNC Mode

25.1 Introduction

NOTE

In MPC832x the BISYNC protocol requires a software patch from Freescale. Please contact an FAE.

A UCC can be configured as a BISYNC controller. A BISYNC controller communicates using the byte-oriented BISYNC protocol. This protocol was developed by IBM for use in networking products. There are three classes of BISYNC frames—transparent[1], nontransparent with header, and nontransparent without header, shown in Figure 25-1. 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 25-1. 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 (Cyclic Redundancy Check) format if 8-bit characters are used; and 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 unique in that TBNR underrun (Transmit Buffer Not Ready and the previous TxBD was with “Last=0”) does not cause an underrun error. If a TBNR underrun event occurs, a synchronization pattern is sent until data is again ready. In nontransparent operation, the receiver discards additional

1. The transparent frame in BISYNC is not related to the transparent mode, discussed in Chapter 28, “Transparent Controller”

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 PowerQUICC II Pro or PowerQUICC III.

A UCC 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 with the non-multiplexed serial interface (NMSI). The controller has separate transmit and receive sections whose operations are asynchronous with the CPU, and either synchronous or asynchronous with other UCCs.

25.1.1 BISYNC Block Diagram

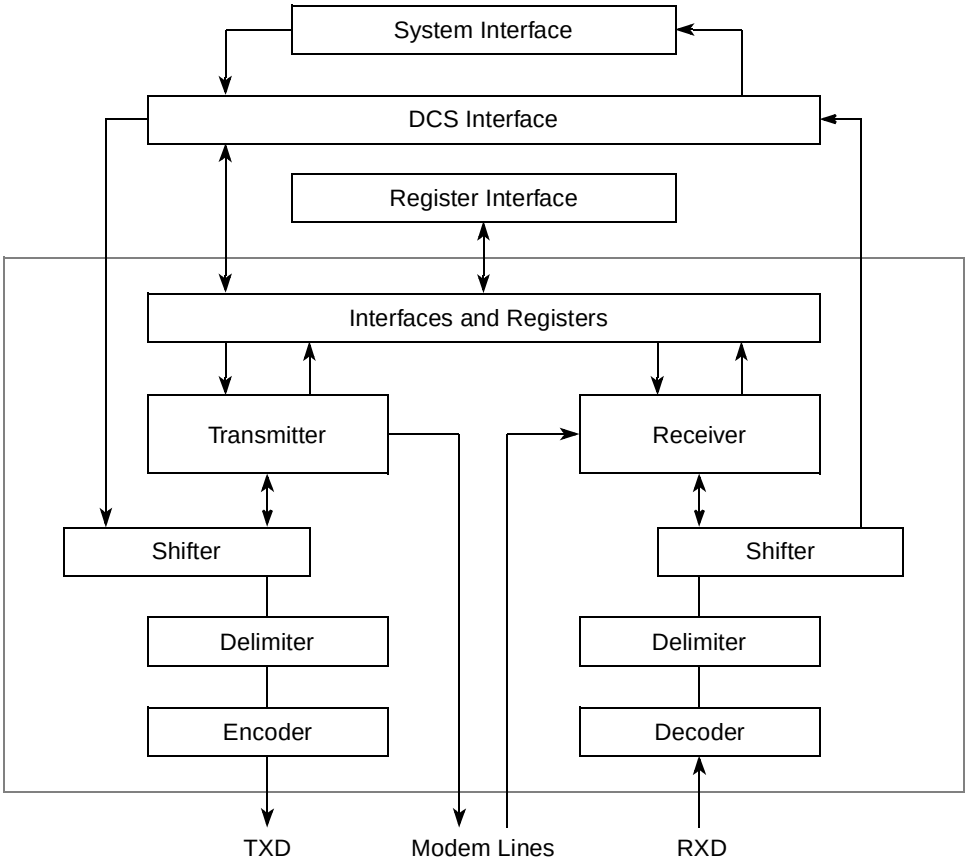


Figure 25-2. BISYNC Block Diagram

25.1.2 Features

The following list summarizes the main features of a UCC when used as a BISYNC controller:

- 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

25.1.3 Modes of Operation

The UCC as a BISYNC controller works with two major modes:

- Transparent mode
- Nontransparent mode

25.2 External Signal Descriptions

The UCC as a BISYNC controller has five signals which include the data and modem line signals. These are described in Table 25-1 below:

Table 25-1. Signal Properties

Name	Function	I/O	Reset
$\overline{\text{CD}}$	Carrier detect	I	1
$\overline{\text{CTS}}$	Clear to send	I	1
$\overline{\text{RTS}}$	Ready to send	O	1
RXD	Receive data line	I	—
TXD	Transmit data line	O	1

25.2.1 Detailed Signal Descriptions

When the GUMR_Lx[DIAG] bit field is programmed for normal operation, $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are controlled by the UCC. In the following sections, it is assumed that the GUMR_Lx[TCI] bit is zero, implying normal transmit clock operation.

Table 25-2. Interface A—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{CD}}$	I	Carrier detect should encapsulate data in certain configurations	
		Timing	Assertion—according to TCLK (transmitter clock). Negation—according to TCLK (transmitter clock).

Table 25-2. Interface A—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{CTS}}$	I	$\overline{\text{CTS}}$ can be used to control reception and transmission in the same manner as the synchronous protocols.	
		Timing	Assertion—$\overline{\text{CTS}}$ transitions must occur while the Tx clock is low. Negation—$\overline{\text{CTS}}$ transitions must occur while the Tx clock is low.
$\overline{\text{RTS}}$	O	Ready To Send signal asserted, when UCC has data to transmit.	
		Timing	Assertion—the delay between $\overline{\text{RTS}}$ and data is 0 bit times.
TXD	O	Serial data out line	
		Timing	Assertion/Negation—according to SERIAL CLK TCLK for transmitter.
RXD	I	Serial data In line	
		Timing	Assertion/Negation—according to SERIAL CLK RCLK for receiver.

25.3 Memory Map/Register Definition

25.3.1 Overview

Table 25-3. UCC BISYNC Register Summary

Offset [1]	Register	Access	Reset Value	Section/Page
0x0	General UCC Mode Register (GUEMR)	R/W	0x0000_0000	23.2.2/23-3
0x8	UPSMR BISYNC Mode Register	R/W	0x0000_0000	25.3.2.5/25-9
0xC	UCC Data Synchronization Register (UDSR)	R/W	0x7E_7E	23.2.3/23-7
0x10	UCCE UCC BISYNC Event Register	R/W	0x0000_0000	25.3.2.8/25-14
0x14	UCCM UCC BISYNC Mask Register	R/W	0x0000_0000	25.3.2.8/25-14

1 From UCC*x* base address

25.3.2 Register Descriptions

25.3.2.1 UCC BISYNC Parameter RAM

For BISYNC mode, the protocol-specific area of the UCC parameter RAM is mapped as shown in Table 25-4. Fields shown in bold text must be initialized by the user.

Table 25-4. UCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**CRCC**	Word	CRC constant temp value

Table 25-4. UCC BISYNC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
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 QUICC Engine module 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 TBNR 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 25.3.2.3, “BISYNC SYNC Register (BSYNC)”.
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 25.3.2.4, “UCC BISYNC DLE Register (BDLE)”.
0x42	**CHARACTER1**	Hword	Control character 1–8. These values represent control characters that the BISYNC controller recognizes. See Section 25.3.2.2, “UCC BISYNC Control Character Recognition (in the parameter RAM)”.
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 25.3.2.2, “UCC BISYNC Control Character Recognition”.
0x54	—	Word	Reserved. Should be cleared.
0x58	—	Word	Reserved. Should be cleared.
0x5c	—	Word	Reserved. Should be cleared.
0x60	—	Word	Reserved. Should be cleared.
0x64	—	Word	Reserved. Should be cleared.
0x68	—	Word	Reserved. Should be cleared.
0x6C	—	Byte	Reserved. Should be cleared.
0x6D-0x100	—	—	Reserved. Should be cleared.

Note:

1 From UCC*x* page base address.

GUMR[MODE] determines the protocol for each UCC. The SYN1-SYN2 synchronization characters are programmed in the DSR (see Section 23.2.3, “UCC Data Synchronization Register (UDSR)”). 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 inspects the data on a per-byte basis and interrupts the CPU each time a byte is received.
- The controller can be programmed so software handles the first 2 or 3 bytes. The controller directly handles subsequent data without interrupting the CPU.

25.3.2.2 UCC 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 1 byte. In single-byte buffers, each byte can be easily inspected, so control character recognition should be disabled.

The control character table 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. The control character table and RCCM are shown in Figure 25-3.

Offset[1]	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 25-3. Control Character Table and RCCM

1 From UCC*x* page base address

Table 25-5 describes control character table and RCCM fields.

Table 25-5. Control Character Table and RCCM Field Descriptions

Offset[1]	Bits	Name	Description
0x42–0x50	0	E	End of table 0 This entry is valid. The lower 8 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 immediately closed. 1 The character is written into the receive buffer. The receiver waits for 1 LRC or 2 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.

Note:

1 From UCC*x* page base address

25.3.2.3 BISYNC SYNC Register (BSYNC)

BSYNC, shown in Figure 25-4, defines BISYNC stripping and SYNC character insertion. When a TBNR 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.

Offset[1]	0	1	2	3	4	5	6	7	8 … 15
0x3E	V	DIS	0	0	0	0	0	0	SYNC

Figure 25-4. BISYNC SYNC (BSYNC)

1 From UCC*x* page base address

Table 25-6 describes BSYNC fields.

Table 25-6. 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

25.3.2.4 UCC BISYNC DLE Register (BDLE)

BDLE, shown in Figure 25-5, defines the BISYNC stripping and insertion of DLE characters. When a TBNR 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.

Offset[1]	0	1	2	3	4	5	6	7	8 … 15
0x40	V	DIS	0	0	0	0	0	0	DLE

Figure 25-5. BISYNC DLE (BDLE)

1 From UCC*x* page base address

Table 25-7 describes the BDLE fields.

Table 25-7. 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 memory. The BCS is calculated only on the second DLE. DLE-SYNC sequence. Both characters are written to 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.

Table 25-7. BDLE Field Descriptions (continued)

Bits	Name	Description
2–7	—	All zeros
8–15	DLE	DLE character

25.3.2.4.1 Sending and Receiving the Synchronization Sequence

The BISYNC channel can be programmed to send and receive a synchronization pattern defined in the DSR. GUMR_H[SYNL] defines the pattern length, as shown in Table 25-8. 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 25-8. Receiver SYNC Pattern Lengths of the DSR

GUMR_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															

25.3.2.5 BISYNC Mode Register (UPSMR)

UPSMR is shown in Figure 25-6. UPSMR[NOS, RBCS, RTR, RPM, TPM] can be modified on-the-fly.

Address: UCC_BASE + 0x8 Access: Read/Write

	0–3	4–5	6	7	8	9	10–11	12–13	14–15
R W	NOS	CRC	RBCS	RTR	RVD	DRT	—	RPM	TPM
Reset	All zeros								

Figure 25-6. Protocol-Specific Mode Register for BISYNC (UPSMR)

Table 25-9 describes UPSMR fields.

Table 25-9. UPSMR Field Descriptions

Bits	Name	Description
0–3	NOS	Minimum number of SYN1-SYN2 pairs (defined in DSR) sent between or before messages. 0000 one pair is sent. 0001 two pairs are sent. . . . 1111 16 pairs are sent. The entire pair is always sent, regardless of how GUMR[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 that 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 8-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 UCC 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 UCC 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. **Note:** If DRT = 1, GUMR_H[CDS] should be cleared unless both of the following are true: the same clock is used for TCLK and RCLK, and $\overline{\text{CTS}}$ either has synchronous timing or is always asserted.
10–11	—	Reserved, should be cleared

Table 25-9. UPSMR 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)

25.3.2.6 UCC BISYNC Receive BD (RxBD)

The QUICC Engine module 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

Figure 25-7 shows the UCC BISYNC RxBD.

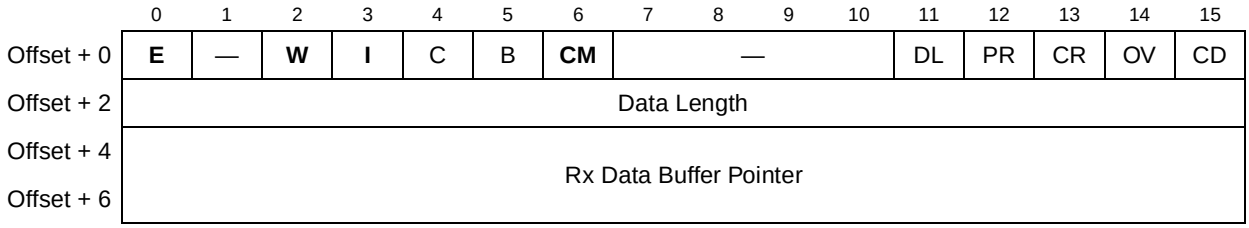


Figure 25-7. UCC BISYNC RxBD

Table 25-10 describes UCC BISYNC RxBD status and control fields. Fields shown in bold type must be initialized by the user.

Table 25-10. UCC 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 cpu can read or write any fields of this RxBD. The QUICC Engine module does not use this BD as long as the E bit is zero. 1 The buffer is not full. The QUICC Engine module controls this BD and buffer. The cpu should not update this BD.
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 QUICC Engine module 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 multi-user RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is used 1 UCCE[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 byte 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 BCS.
6	**CM**	Continuous mode. 0 Normal operation 1 The QUICC Engine module does not clear E after this BD is closed; the buffer is overwritten when the QUICC Engine module accesses this BD next. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7–10	—	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[RBCS] within eight serial clocks, the user can exclude the current character from the massage 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, $\overline{\text{CD}}$, is negated during frame reception.

Data length and buffer pointer fields are described in Section 23.3, "UCC Buffer Descriptors (BDs)." Data length represents the number of octets the QUICC Engine module 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.

25.3.2.7 UCC BISYNC Transmit BD (TxBD)

The QUICC Engine module arranges data to be sent on a UCC channel in buffers referenced by the channel TxBD table. The QUICC Engine module uses BDs to confirm transmission or indicate errors so the CPU knows buffers have been serviced. The QUICC Engine module configures status and control bits before transmission, but the QUICC Engine module sets them after the buffer is sent.

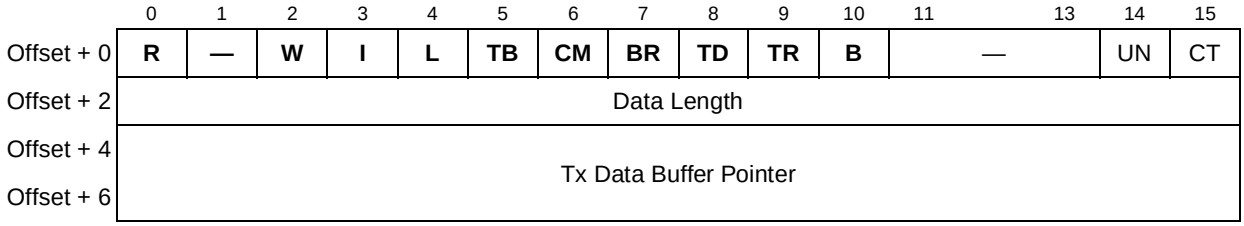


Figure 25-8. UCC BISYNC Transmit BD (TxBD)

Table 25-11 describes UCC BISYNC TxBD status and control fields. Fields shown in bold type must be initialized by the user.

Table 25-11. UCC 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 QUICC Engine module 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 QUICC Engine module 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 multi-user RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 UCCE[TXB] or UCCE[TXE] is set after the QUICC Engine module 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 GUMR[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 QUICC Engine module does not clear R after this BD is closed, so the buffer is resent when the QUICC Engine module next accesses this BD. However, R is cleared if an error occurs during transmission, regardless of how CM is set.

Table 25-11. UCC BISYNC TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
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.
9	**TR**	Transparent mode 0 The transmitter enters and stays in normal mode after sending the buffer. The transmitter automatically inserts SYNCs if a TBNR underrun condition occurs. 1 The transmitter enters or stays in transparent mode after sending the buffer. It automatically inserts DLE–SYNC pairs if a TBNR 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 UPSMR[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 QUICC Engine module writes UN after it sends the associated buffer.
15	CT	$\overline{\text{CTS}}$ lost. The QUICC Engine module 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 23.3, “UCC Buffer Descriptors (BDs).” Although it is never modified by the QUICC Engine module, data length should be greater than zero. The QUICC Engine module writes these fields after it finishes sending the buffer.

25.3.2.8 BISYNC Event Register (UCCE)/BISYNC Mask Register (UCCM)

The BISYNC controller uses UCCE to report events recognized by the BISYNC channel and to generate interrupts, as shown in Figure 25-9. When an event is recognized, the controller sets the corresponding UCCE bit. Interrupts are enabled by setting, and masked by clearing the equivalent bits in UCCM. UCCE bits are reset by writing ones; writing zeros has no effect. Unmasked bits must be reset before the QUICC Engine module negates the internal interrupt request signal.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM) Access: Read/Write

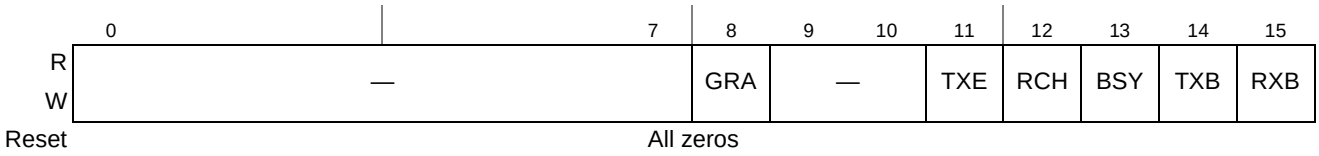


Figure 25-9. BISYNC Event Register (UCCE)/BISYNC Mask Register (UCCM)

Table 25-12 describes UCCE and UCCM fields.

Table 25-12. UCCE/UCCM Field Descriptions[1]

Bits	Name	Description
0–7	—	Reserved, should be cleared. Refer to note 1 below.
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. Refer to note 1 below.
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 QUICC Engine module closes the receive buffer on the BISYNC channel.

1 Reserved bits in the UCCE should not be masked in the UCCM register.

25.4 Functional Description

25.4.1 UCC BISYNC Channel Frame Transmission

The BISYNC transmitter is designed to work with almost no CPU intervention. When the transmitter is enabled, it starts sending SYN1-SYN2 pairs in the data synchronization register (DSR) or idles as programmed in the GUMR. 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 GUMR[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 GUMR[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 TBNR 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.

25.4.2 UCC BISYNC Channel Frame Reception

Although the receiver is designed to work with almost no CPU 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 25.3.2.2, "UCC BISYNC Control Character Recognition."

When enabled, the receiver enters the hunt mode where the data is shifted into the receiver shift register 1 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, the hunt mode is terminated and character assembly begins. The controller is character-synchronized and performs SYNC stripping and message reception. It reverts to the 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).

25.4.3 UCC BISYNC Commands

Transmit and receive commands are issued to the QUICC Engine module command register (CPCR). Transmit commands are described in Table 25-13.

Table 25-13. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GUMR, 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 DCS 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.
GRACEFUL STOP TRANSMIT	Stops transmission after the current frame finishes sending or immediately if there is no frame being sent. UCCE[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.

Table 25-13. Transmit Commands (continued)

Command	Description
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 UCCM. 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 25-14.

Table 25-14. 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 UCCM, 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.
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.

25.4.4 Handling Errors in the UCC BISYNC

The controller reports message transmit and receive errors using the channel BDs, error counters, and the UCCE. Modem lines can be directly monitored via the parallel port pins. Table 25-15 describes transmit errors.

Table 25-15. Transmit Errors

Error	Description
Transmitter underrun	The channel stops sending the buffer, closes it, sets TxBD[UN], and generates a TXE 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 25-16 describes receive errors.

Table 25-16. Receive Errors

Error	Description
Overrun	The controller maintains a receiver DCS for receiving data. The QUICC Engine module 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 DCS. If an Rx DCS 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. When GUMR[SYNL] = 1x (8-bit sync or 16-bit sync), and $\overline{\text{CD}}$ lost during syncs or first data byte, data doesn't transferred to Rx FIFO.
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 eight 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.

25.5 Initialization Information

25.5.1 Programming the UCC 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 2 to 3 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 UCCM[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 UPSMR[RTR] or issue a RESET BCS CALCULATION command. For example, if a DLE-STX is received, enter transparent mode. By setting the appropriate UPSMR bit, the controller strips the leading DLE from DLE-character sequences. Thus, control characters are recognized only when they follow a DLE character. UPSMR[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 UPSMR[RBCS] is not needed because the controller automatically excludes SYNCs and leading DLEs.

After the type of block is recognized, UCCE[RCH] should be masked. The CPU 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 25-17, the control character table should be set to recognize the end of the block.

Table 25-17. 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 UCCM[RCH] to reenable interrupts for each byte of data received.

Chapter 26
UCC for Fast Protocols

NOTE

Read Chapter 22, "Unified Communications Controllers (UCCs)," before reading this chapter.

The QUICC Engine block unified communications controller (UCC) implements a wide range of protocols and interfaces that are divided into fast and slow protocols. This chapter provides a general overview of the UCC when used for fast protocols and the common programming model. The fast protocols include ATM over UTOPIA L2 and high-speed serials (Ethernet, HDLC, HDLC bus, Transparent). Slow protocols (UART and asynchronous HDLC, BISYNC, and QMC) are described in their respective chapters. For a detailed description of the feature set and the protocol-specific programming model, refer to their respective UCC chapter.

UCC key features for fast protocols include the following:

- Supports the following interfaces
  - UTOPIA L2 at 50 MHz (assuming QUICC Engine frequency above 100 MHz[1]).
  - MII/RMII (assuming QUICC Engine frequency above 100 MHz).
  - QUICC Engine frequency
  - High speed serial interface[2] with modem control.
  - Serial or nibble-parallel data interface through TDM[3].
- Supports ATM, Ethernet and HDLC protocols. Assuming a 200-MHz QUICC Engine clock, the UCC supports the following:
  - Full 10/100 Mbps, full-duplex/half-duplex Ethernet/IEEE 802.3x/VLAN through MII/RMII
  - Full OC-3 rate full-duplex ATM AAL5 segmentation and reassembly (SAR) through UTOPIA L2
  - Up to 70 Mbps HDLC/ HDLC bus and/or totally transparent protocols data rates
- UCC clock 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
- Uses bursts to improve bus usage
- Multibuffer data structure for received and transmit
- Buffers and buffer descriptors (BDs) may reside anywhere in system memory.
- Programmable size-virtual FIFO buffers

1. Note that this is the minimal QUICC Engine frequency for H/W operation. The actual QUICC Engine frequency to meet the protocol performance target will be higher.
2. The QUICC Engine frequency must be higher than the data bit-rate multiplied by 8/3.
3. The QUICC Engine frequency must be higher than the data bit-rate multiplied by 24.

- Fully transparent option for one half of the UCC (receiver/transmitter) while HDLC/SDLC protocols execute on the other half (transmitter/receiver)
- Echo and local loopback modes for testing
- Routing to any TDM through the TSA in serial or nibble data port size

26.1 Overview

The UCC protocol is chosen by programming the mode bits in the GUMR register (see Section 26.4.2.1, "GUMR in Fast Mode"). For a fast protocol mode such as ATM, Ethernet, HDLC/ HDLC bus or Transparent, the UCC must first be configured to work in the fast mode by configuring GUEMR register.

FCC compatibility is provided for the following protocols:

- HDLC
- Transparent

Subsequent chapters in this book describe extended features. UCC initialization code is slightly different from FCC or SCC code. Differences are emphasized throughout the manual.

NOTE: Backward-Compatibility

The UCC is backward-compatible to the FCC in HDLC or Transparent modes.

In 10/100BaseT Ethernet mode, some of the mode bits in the FCC programming model moved to other registers in the UCC Ethernet Controller (UEC). The data structures (buffer descriptors) are the same as in the MPC8260 FCC (and TSEC). The UCC parameter RAM is very similar to the FCC in Ethernet mode. The manual specifies the differences. In ATM mode, the programming model for configuring the UTOPIA bus interface is described in the UPC chapter.

The UCC block diagram is shown in Figure 26-1.

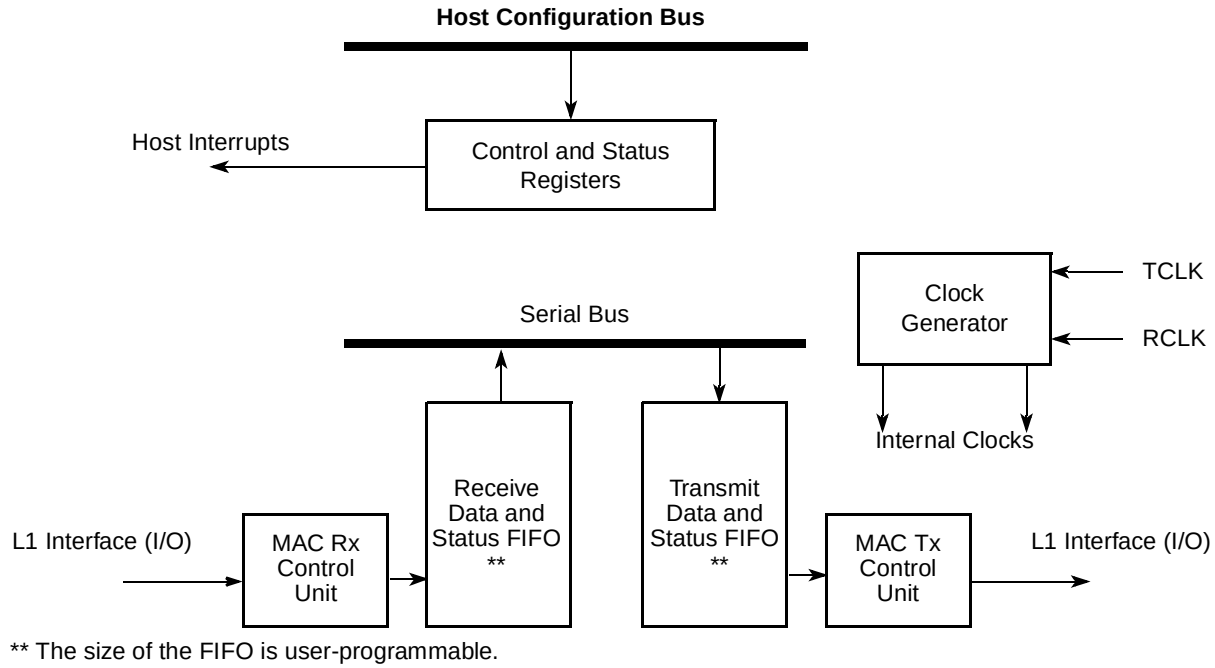


Figure 26-1. UCC Block Diagram

26.2 UCC Virtual FIFOs

NOTE: Backward Compatibility

Users who are familiar with the PowerQUICC II should note the following change in the UCC compared with the FCC.

Fast protocols require larger FIFO depth than slow ones. Sufficient FIFO size is important for increasing the QUICC Engine block's performance by eliminating overrun and underrun conditions. Each protocol has an optimized FIFO size that depends not only on the bit rate, but also on other parameters such as packet or frame size. The UCC, when configured for fast protocols, extends the HW FIFO by using the QUICC Engine block's internal MURAM. This extension is called virtual FIFO or VFIFO, because its size and location within the MURAM are programmable according to the specific protocol. The FIFO as shown in Figure 26-1 is constructed of a real HW FIFO embedded in the UCC and its extension to a virtual FIFOs in the QUICC Engine block's MURAM. This flexible FIFO structure enables the QUICC Engine block to sustain maximum data rates by allocating larger FIFO memory when required.

The size of the virtual FIFO is user-programmable (see Section 26.5, "Fast Protocol FIFO Configuration Registers"). The actual size that is needed depends on a number of factors, especially the following:

- Maximum packet size
- Protocols running on the UCC
- Memory bus latency

26.3 UCC Parameter RAM for Fast Protocols

The QUICC Engine block maintains a section of MURAM called the parameter RAM, which contains many parameters for the operation of the UCCs. UCC base addresses are described in Section 22.3.2, "General UCC Extended Mode Register (GUEMR)."

The UCC Parameter RAM base addresses are programmable using the CECR command register, see Section 19.3.1, "QUICC Engine Command Register (CECR)."

Some parameter RAM values must be initialized before the UCC is enabled. The QUICC Engine block initializes or writes other values. Once initialized, most parameter RAM values need not be accessed by user software, because most activity centers around the TxBDs and RxBDs rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled (that is, 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 RX BD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.

NOTE: Backward-Compatibility

After power-on reset, UCC1–UCC3 are backward-compatible to FCC1–FCC3.

In the 82xx family of devices, the parameter RAM for each serial controller is located at a fixed address in the internal memory space for the device. In the QUICC Engine architecture there is a default setting for the parameter RAM pages (see Section 22.3.2, "General UCC Extended Mode Register (GUEMR)"), but the user may relocate the pages to allow improved optimization of the memory space.

26.4 Programming Model

The following sections discuss the programming model.

26.4.1 UCC Memory Map for Fast Protocols

Table 26-1. UCC Memory Map (Fast Mode)

Address[1]	Use	Size	Access
0x00	General UCC mode register (GUMR)	4	R/W
0x04	UCC protocol specific mode register (UPSMR)[2, 3]	4	R/W
0x08	UCC transmit on demand register (UTODR)	2	R/W
0x0C	UCC data synchronization register (UDSR)	2	R/W

Table 26-1. UCC Memory Map (Fast Mode) (continued)

Address[1]	Use	Size	Access
0x10	UCC event register (UCCE)	4	R/W
0x14	UCC mask register (UCCM)	4	R/W
0x18	UCC status register (UCCS) [3]	1	R
	Virtual FIFO Registers		
0x20	UCC receive virtual FIFO base (URFB)	4	R/W
0x24	UCC receive virtual FIFO size (URFS)	2	R/W
0x28	UCC receive virtual FIFO emergency threshold (URFET)	2	R/W
0x2A	UCC receive virtual FIFO Special emergency threshold (URFSET)	2	R/W
0x2C	UCC transmit virtual FIFO base (UTFB)	4	R/W
0x30	UCC transmit virtual FIFO size (UTFS)	2	R/W
0x34	UCC transmit virtual FIFO emergency threshold (UTFET)	2	R/W
0x38	UCC transmit virtual FIFO, transmit threshold (UTFTT)	2	R/W
	Timer/Counter Registers		
0x3C	UCC transmit polling timer (UTPT)[4]	2	R/W
0x40	UCC retry counter register (URTRY)	4	R
0x44–0x7F	Reserved	12	
	Extended mode register		
0x90	General UCC extended mode register (GUEMR)	1	R/W
0x94–0xFF	Reserved	12	
0x100–0x1BF	Ethernet MAC specific registers		R/W
0x1C0–0x1FF	Reserved	12	

Note:

1 Offset from UCCx base

2 In Ethernet RMII mode include also Loopback and CLK that are configured in the GUMR, see Section 26.4.2.1, “GUMR in Fast Mode.”

3 Protocol specific register, described in the specific protocols’ chapters.

4 See Chapter 22, “Unified Communications Controllers (UCCs).”

26.4.2 UCC Registers in Fast Mode

The following sections discuss UCC registers in fast mode.

26.4.2.1 GUMR in Fast Mode

GUMR configures the protocol and interface of the UCC and is compatible with the PowerQUICC II general FCC mode register (GFMR). The GUMR mode defines the programming model for the UCC and the meaning of the remaining bits of this register. In this chapter the following fast protocols are described:

- HDLC
- Transparent
- Ethernet
- ATM

NOTE: Backward-Compatibility

GUMR[14–15] are modes taken from the PowerQUICC II SCC. For backward-compatibility with the PowerQUICC II FCC, they should be cleared.

Offset UCC base + 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	13	14	15
R W	DIAG		TCI	TRX	TTX	CDP	CTSP	CDS	CTSS	—	—	TXSY	RSYN
Reset	All zeros												

	16	17	18	19	20	21	22	23	24	25	26	27	28	31
R W	SYNL		RTSM	RENC		REVD	TENC		TCRC		ENR	ENT	MODE	
Reset	All zeros													

Figure 26-2. General UCC Mode Register (Fast Protocols)

Table 26-2. GUMR (in Fast Mode) 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 UCC uses the modem signals ($\overline{CD}$ and $\overline{CTS}$) to enable and disable transmission and reception automatically. Timings are shown in Section 22.5.1, "Synchronous Protocols" and in Section 22.5.2, "Asynchronous Protocols". 01 Local loopback mode—The transmitter output is connected internally to the receiver input; the receiver and 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. RTS can be disabled in the appropriate parallel I/O register. In TDM modes, L1TXDx and L1RQx can be programmed either to be asserted normally or to remain inactive by programming the serial interface mode register (SIMODE). The transmitter and receiver must use the same clock source; thus, the same BRG or the same external CLKx signal can be used. Separate CLKx signals can be used as long as they are connected to the same external clock source. **Note:** If external loopback is preferred, DIAG should be set for normal operation and TXD and RXD should be connected externally. Clocks can be generated externally or they can be generated internally by a baud-rate generator. The user can physically connect the appropriate control signals (RTS connected to CD, and CTS grounded), or the parallel port register can be used to cause CD and CTS to be asserted to the UCC permanently. 10 Automatic echo mode—The channel automatically retransmits received data, using the receive clock provided. The receiver operates normally and receives data if CD is asserted. The transmitter simply transmits received data. In this mode, CTS is ignored. The echo function can also be accomplished in software by receiving buffers from a UCC, linking them to TxBDs and transmitting them back out of that UCC. **Note:** For correct ECHO operation the Rx and Tx clocks should be identical. 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. **Note:** For correct ECHO operation the Rx and Tx clocks should be identical.
2	TCI	Transmit clock invert 0 Normal operation (For Ethernet and ATM the transmitter is triggered by the rising edge of the UCC Tx clock, for HDLC and Transparent the UCC inverts the reference clock so it can clock data out one-half clock earlier). 1 The UCC inverts the internal transmit clock. (For HDLC and Transparent protocols data is clocked out on the rising edge of the UCC Tx clock). **Note:** It is recommended to set TCI when the reference clock frequency is above 20 MHz (also in internal loopback) **Note:** This bit should be cleared if this UCC is used with the TSA. **Note:** TCI does not apply for Ethernet and ATM protocols.
3	TRX	Transparent receiver. The UCCs 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 mode lets the user implement unique applications such as a UCC 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:** If full-duplex, totally transparent operation for a UCC is obtained by setting both TTX and TRX. Attempting to operate a UCC with Ethernet or ATM on its transmitter and transparent operation on its receiver causes erratic behavior. In other words, unless MODE = HDLC, TTX must equal TRX.

Table 26-2. GUMR (in Fast Mode) Register Field Descriptions (continued)

Bits	Name	Description
4	TTX	Transparent transmitter. The QUICC Engine UCCs 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 mode lets the user implement unique applications, such as configuring a UCC 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:** If full-duplex, totally transparent operation for a UCC is obtained by setting both TTX and TRX. Attempting to operate a UCC with Ethernet or ATM on its transmitter and transparent operation on its receiver causes erratic behavior. In other words, unless MODE = HDLC, TTX must equal TRX.
5	CDP	$\overline{CD}$ pulse 0 Normal operation (envelope mode). $\overline{CD}$ should envelope the frame; the negation of $\overline{CD}$ while receiving causes a $\overline{CD}$ lost error. 1 Pulse mode. When $\overline{CD}$ is asserted (high to low transition), synchronization has been achieved, and further transitions of $\overline{CD}$ do not affect reception. **Note:** This bit must be set if this UCC is used with the TSA in transparent mode.
6	CTSP	$\overline{CTS}$ pulse 0 Normal operation (envelope mode). $\overline{CTS}$ should envelope the frame; the negation of $\overline{CTS}$ while transmitting causes a $\overline{CTS}$ lost error. 1 Pulse mode. $\overline{CTS}$ is asserted when synchronization is achieved; further transitions of $\overline{CTS}$ do not affect transmission. **Note:** This bit must be set if this UCC is used with the TSA, except for ISDN D-channel in IDL circuits, for which normal (envelope) mode must be selected. **Note:** This bit must be cleared if this UCC is used in internal loopback.
7	CDS	$\overline{CD}$ sampling 0 The $\overline{CD}$ input is assumed to be asynchronous with the data. The UCC synchronizes it internally before data is received. (This mode is illegal 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 mode is useful when connecting CEs in transparent mode because it allows the $\overline{RTS}$ signal of one QUICC Engine block to be connected directly to the $\overline{CD}$ signal of another QUICC Engine block. **Note:** This bit must be set if this UCC is used with the TSA. **Note:** This bit must be set if this UCC is used in internal loopback.
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 UCC, data is sent after no more than two serial clock delays. 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 QUICC Engine block in transparent mode because it allows the $\overline{RTS}$ signal of one QUICC Engine block to be connected directly to the $\overline{CTS}$ signal of another QUICC Engine block. **Note:** This bit must be set if this UCC is used with the TSA.
9–13	—	Reserved. Should be cleared.

Table 26-2. GUMR (in Fast Mode) Register Field Descriptions (continued)

Bits	Name	Description
14	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 UCC. Assuming $\overline{CTS}$ is asserted, transmission begins 8 clocks after the receiver starts receiving data. **Note**: Should be cleared in all protocols except for transparent operation. Can be set only in serial (1 bit data width) interface. **Note**: For backward compatibility with the PowerQUICC II FCC, this bit should be cleared.
15	RSYN	Receive synchronization timing (totally transparent mode only). 0 Normal operation 1 If CDS = 1, $\overline{CD}$ should be asserted on the second bit of the Rx frame rather than on the first. **Note**: Should be cleared in all protocols except for transparent operation. Can be set only in serial (1 bit data width) interface. **Note**: For backward compatibility with the PowerQUICC II FCC, this bit should be cleared. **Note:** This bit must be cleared if this UCC is used in internal loopback.
16–17	SYNL	Sync length (transparent mode only). Determines the operation of a UCC receiver configured for totally transparent operation only. 00 The sync pattern in the UDSR 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 UDSR. The negation of $\overline{CD}$ causes CD lost error. Transmitter is not affected from this mode. 11 16-bit sync. The receiver synchronizes on a 16-bit sync pattern stored in the UDSR. The Negation of $\overline{CD}$ causes CD lost error. Transmitter is not affected from this mode.
18	RTSM	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 UCC 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
21	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation 1 The totally transparent channels on this UCC (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

Table 26-2. GUMR (in Fast Mode) Register Field Descriptions (continued)

Bits	Name	Description
24–25	TCRC	Transparent CRC (totally transparent channel only). Selects the type of frame checking provided on the transparent channels of the UCC (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 UCC. 0 Reset the UCC receiver. The receiver is disabled and any data in the receive FIFO buffer is lost. 1 The receiver is enabled. **Note:** The UCC provides other tools for controlling reception—the enter hunt mode command, close rx bd command, and RxBD[E]. **Note:** When working in Ethernet mode, set the MACCFG1 bit 29 (Enable Receiver) also.
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this UCC. 0 Reset the UCC transmitter. 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 TxFIFOs is flushed. 1 The transmitter is enabled. **Note:** The UCC provides other tools for controlling transmission besides the ENT bit—the stop transmit, graceful stop. **Note:** When working in Ethernet mode need to Set also MACCFG1 bit 31(Enable Transmitter)
28–31	MODE	Channel protocol mode: 0000 HDLC 0001 Reserved 0010 Reserved (QMC) 0011 Reserved 0100 Reserved (UART) 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved (BISYNC) 1001 Reserved 1010 ATM 1011 Reserved 1100 Ethernet 1101 Reserved 1110 Reserved 1111 Reserved

26.4.3 UCC Transmit Polling Timer (UTPT)

Refer to Section 22.3.3, “UCC Transmit Polling Timer (UTPT).”

26.4.4 UCC Transmit On Demand Register (UTODR)

Refer to Section 22.3.4, “UCC Transmit-On-Demand Register (UTODR).”

26.4.5 UCC Data Synchronization Register (UDSR)

NOTE: Backward-Compatibility

The UDSR is backward-compatible with the FCC FDSR for HDLC and transparent protocols. It is not used for Ethernet protocol in the QUICC Engine block.

Each UCC has a UDSR to specify the pattern used in the frame synchronization procedure of the synchronous protocols. In the totally transparent protocol, the UDSR should be programmed with the preferred SYNC pattern. At reset, UDSR defaults to 0x7E7E (two HDLC flags) and need not be written for HDLC mode. The UDSR contents are always sent lsb first.

Offset UCC base + 0x0C — Access: Read/Write

	0							7	8							15
R W	SYN2								SYN1							
Reset	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0

Figure 26-3. UCC Data Synchronization Register (Fast Protocols)

26.4.6 Bus Mode Registers (RBMR and TBMR)

There are separate bus mode registers for Rx buffers (RBMR) and for Tx buffers (TBMR). On the MPC826x devices, these were called function code registers, RFCR and TFCR. The bus mode registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 26-4 shows the register format. Boldfaced entries must be initialized by the user.

Offset UCC base + 0x04 (RBMR*x*)
UCC base + 0x05 (TBMR*x*) — Access: Read/Write

	0	1	2	3	4	5	6	7
R W	—		**GBL**	**BO**		**CETM**	**DTB**	**BDB**
Reset	All zeros							

Figure 26-4. Bus Mode Registers (RBMR and TBMR)

Table 26-3 describes RBMR*x*/TBMR*x* fields.

Table 26-3. RBMR*x*/TBMR*x* Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global. Indicates whether the memory operation should be snooped. 0 Snooping on the Coherent System Bus (CSB) is disabled. 1 Snooping on the Coherent System Bus (CSB) is enabled. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”

Table 26-3. RBMR*x*/TBMR*x* Field Descriptions (continued)

Bits	Name	Description
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 and transparent) or at the beginning of the next BD. 00,01,11 Reserved modes. 10 Big-endian byte ordering. As data is sent onto the serial line from the data buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same buffer word. These bits must be set to 10 value.
5	**CETM**	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the “Debug Configuration” subsection of your device reference manual.
6	**DTB**	Indicates on what bus the data is located. 0 On the coherent system bus (CSB) 1 On the QUICC Engine secondary bus. The programming of this bit must be consistent with the System Configuration. See Section 18.1.1, “Data Paths.”
7	**BDB**	Indicates on what bus the BDs are located. 0 On the coherent system bus (CSB). 1 On the QUICC Engine secondary bus. The programming of this bit must be consistent with the System Configuration. See Section 18.1.1, “Data Paths.”

26.4.7 UCC Event Register (UCCE)

Refer to Section 22.3.5, “UCC Event Register (UCCE),” and to the protocol specific chapters of the user manual.

26.4.8 UCC Mask Register (UCCM)

Refer to Section 22.3.6, “UCC Mask Register (UCCM),” and to the protocol specific chapters of the user manual.

26.4.9 UCC Status Register

Refer to Section 22.3.7, “UCC Status Register,” and to the protocol specific chapters of the user manual.

26.5 Fast Protocol FIFO Configuration Registers

The following sections discuss fast protocol FIFO configuration registers.

26.5.1 Receive Virtual FIFO Base Register (URFB)

URFB configures the receive virtual FIFO base address relative to the multi-user RAM base address.

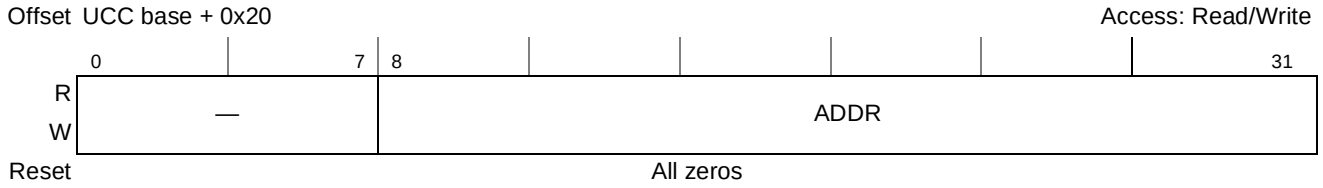


Figure 26-5. Receive Virtual FIFO Base Register (Fast Protocols)

Table 26-4. URFB Register Field Descriptions

Bits	Name	Description
0–7	—	Reserved. Should be cleared by the user.
8–31	ADDR	Receive FIFO base address[1]. Address 0x0 is restricted and can not be used.

1 Address must be aligned to 8 bytes (3 lsb cleared)

26.5.2 Receive Virtual FIFO Size Register (URFS)

URFS configures the receive virtual FIFO size in bytes, allocated to data and status in the multi-user RAM.

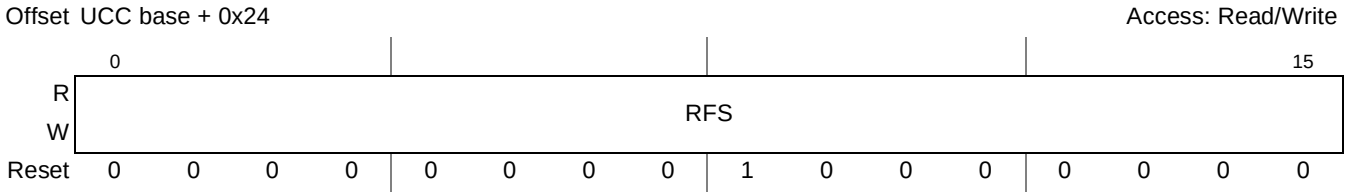


Figure 26-6. Receive Virtual FIFO Size Register (Fast Protocols)

Table 26-5. URFS (in Fast Mode) Register Field Descriptions

Bits	Name	Description
0–15	RFS[1]	Receive virtual FIFO's size (in bytes). The following values are the per-protocol minimum values for RFS: • Fast Ethernet 512 bytes • HDLC/Transparent up to 10 Mbps 128 bytes • HDLC/Transparent above 10 Mbps 256 bytes • ATM OC-3 Rate: 256 bytes

1 Size must be aligned to 8 bytes.

NOTE

The register descriptions contain recommended values for the different protocols, assuming the UCC is used in a system where accessing the external memory is the bottleneck of the system. These values may differ if the memory bus is not the bottleneck.

26.5.3 Receive Virtual FIFO Emergency Threshold (URFET)

URFET configures the lower emergency threshold level of the receive virtual FIFO, which is the minimum amount of bytes in virtual FIFO for which RISC requests for data handling get emergency priority.

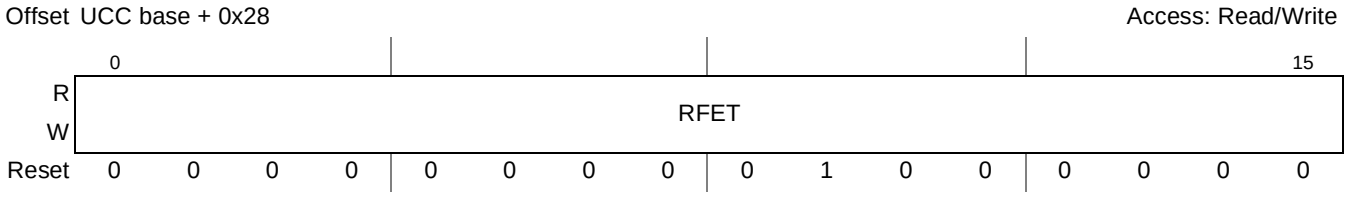


Figure 26-7. Receive Virtual FIFO Emergency Threshold (Fast Protocols)

Table 26-6. URFET (in Fast Mode) Register Field Descriptions

Bits	Name	Description
0–15	RFET[1]	Receive Virtual FIFO Emergency Threshold (in bytes). The recommended value for RFET (for most applications) is 1/2 of the FIFO size. To disable the emergency state, program this threshold to a value which exceeds the FIFOs size.

1 Must be aligned to 8 bytes.

26.5.4 Receive Virtual FIFO Special Emergency Threshold (URFSET)

URFSET configures the upper threshold level of the receive virtual FIFO. The UCC enters a special emergency state when the virtual FIFO's level exceeds RSFET, and exits this state when the virtual FIFO fill level drops below RFET (the normal emergency threshold programmed in URFET register). In Ethernet protocol, during the special emergency state, the UCC transmits pause flow control frames to the link partner. In ATM protocol, during the special emergency state the ATM transmitter may pause transmit.

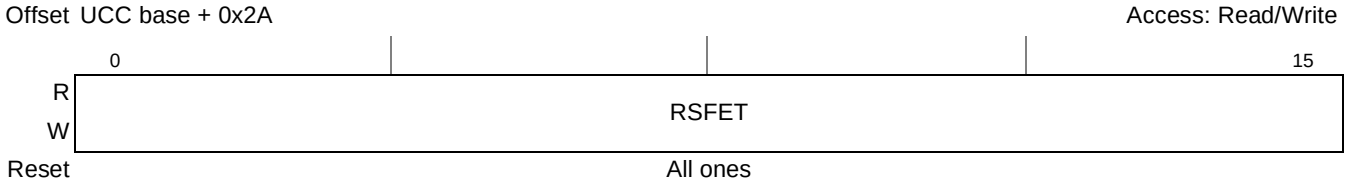


Figure 26-8. Receive Virtual FIFO Special Emergency Threshold (Fast Protocols)

Table 26-7. URFSET (in Fast Mode) Register Field Descriptions

Bits	Name	Description
0–15	RSFET[1]	Receive Virtual FIFO Special Emergency Threshold (in bytes). RSFET should always be larger then RFET. The recommended value for RSFET in order to enable the special emergency state (for most applications) is 3/4 of the FIFO size. To disable the special emergency state, program this threshold to a value which exceeds the FIFOs size (disabled by default at reset).

1 Must be 8 aligned to 8 bytes.

26.5.5 Transmit Virtual FIFO Base Register (UTFB)

UTFB configures the transmit virtual FIFO base address relative to the multiuser RAM base address.

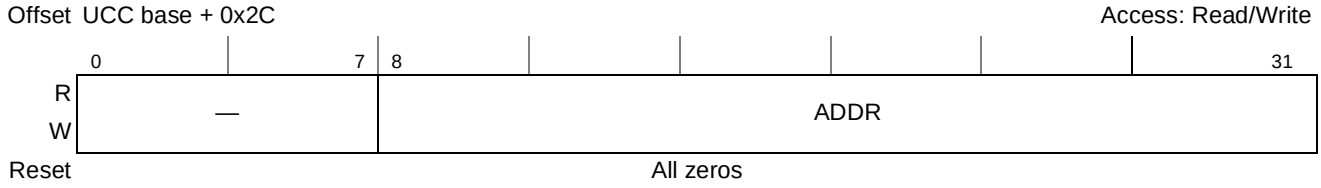


Figure 26-9. Transmit Virtual FIFO Base Register (Fast Protocols))

Table 26-8. UTFB Register Field Descriptions

Bits	Name	Description
0–7	—	Reserved. Should be cleared.
8–31	ADDR	Transmit virtual FIFO base address[1]. Address 0x0 is restricted.

1 Address must be aligned to 8 bytes (3 lsb cleared).

26.5.6 Transmit Virtual FIFO Size Register (UTFS)

NOTE

UTFS configures the transmit virtual FIFO size in bytes, allocated to data and status in the multiuser RAM.

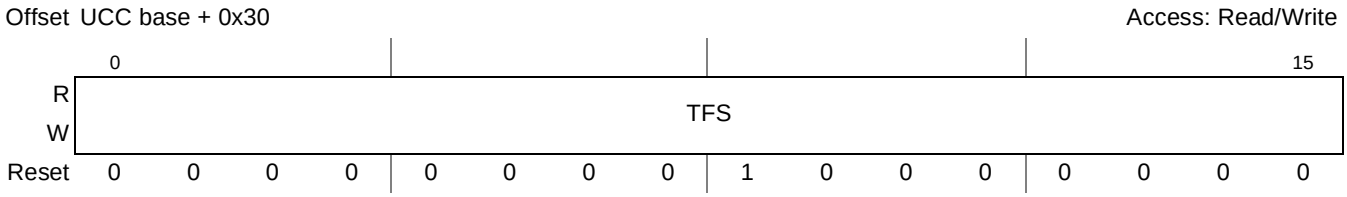


Figure 26-10. Transmit Virtual FIFO Size Register (Fast Protocols)

Table 26-9. UTFS Register Field Descriptions

Bits	Name	Description
0–15	TFS[1]	Transmit Virtual FIFO Size (in Bytes) The following values are the per-protocol minimal values for TFS: • Fast Ethernet 512 bytes • HDLC/Transparent up to 10 Mbps 128 bytes • HDLC/Transparent up to 10 Mbps 256 bytes • ATM OC-3 Rate 256 bytes

Note:

1 Size must be aligned to 8 bytes.

NOTE

The register descriptions contain recommended values for the different protocols, assuming the UCC is used in a system where accessing the external memory is the bottleneck of the system. These values may differ if the memory bus is not the bottleneck.

26.5.7 Transmit Virtual FIFO Emergency Threshold (UTFET)

UTFET configures the threshold level of the transmit emergency state (high priority). When the number of bytes in the virtual FIFO drops below this watermark the UCC transmitter enters emergency state.

Offset UCC base + 0x34 Access: Read/Write

	0								15							
R W	TFET															
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Figure 26-11. Transmit Virtual FIFO Emergency Threshold

Table 26-10. UTFET (in Fast Mode) Register Field Descriptions

Bits	Name	Description
0–15	TFET	Transmit virtual FIFO Emergency Threshold (in bytes). It is recommended to set this value to 1/2 the FIFO size. Additional fine tuning of this register may be needed for performance optimizations.

26.5.8 Transmit Virtual FIFO Transmit Threshold (UTFTT)

UTFTT configures the threshold level of the transmit virtual FIFO. This threshold represents the amount of bytes that should be in the virtual FIFO before transmission starts (transmission will start before

reaching that watermark if the entire frame is in the virtual FIFO). This a condition is checked for each frame.

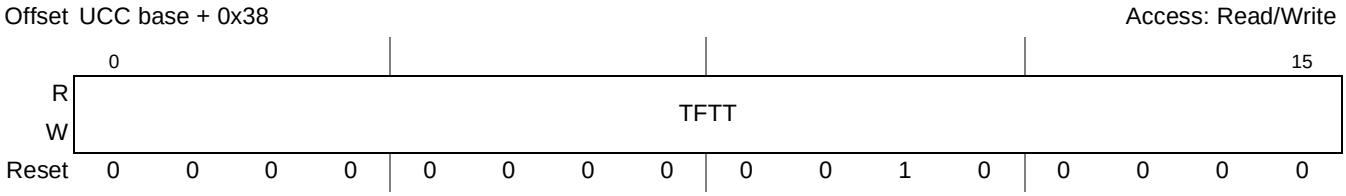


Figure 26-12. Transmit Virtual FIFO Transmit Threshold

Table 26-11. UTFTT (in Fast Mode) Register Field Descriptions

Bits	Name	Description
0–15	TFTT[1]	Transmit virtual FIFO transmit threshold (in bytes). This threshold represents the number of bytes that are placed in the virtual FIFO before transmission starts. The recommended value (for most applications) is 0x40 bytes, or up to 1/4 of the FIFO size. The minimum value allowed is 0x40 bytes, the maximum value is equal to UTFS. (Entire frames or cells which are shorter than UTFTT as transmitted as soon as they are in the virtual FIFO) **Note:** If the UCC is configured to Ethernet Controller, the recommended value (for most applications) is equal to the value programmed in UCC Transmit FIFO Size (UTFS) register.

Note:

1 Must be 8 aligned to 8 bytes.

26.5.9 UCC Transmit Retry Counter (URTRY)

URTRY counts the number of retransmission attempts of the UCC. Retransmission can happen for Ethernet half duplex or for HDLC bus protocols. The retry counter is cleared by writing to it (any value).

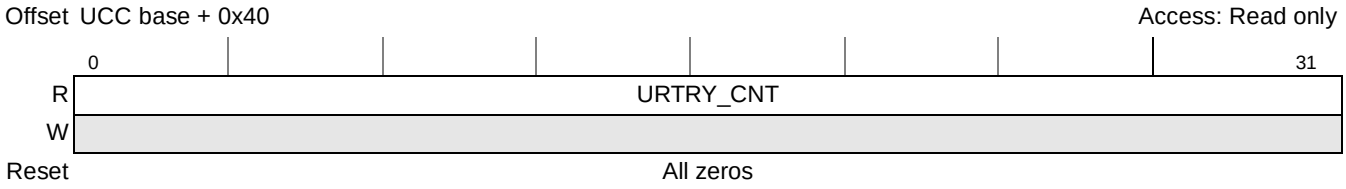


Figure 26-13. DCS Retry Counter Register (URTRY)

Table 26-12. URTRY Register Field Descriptions

Bits	Name	Description
0–31	URTRY_CNT	The number of retransmissions. Read only, cleared by any write to it.

Chapter 27
HDLC Controller

27.1 Introduction

Layer 2 of the seven-layer OSI model is the data link layer (DLL), in which high level data link control (HDLC) is one of the most common protocols. The framing structure of HDLC is shown in Figure 27-10. 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 GUMR[MODE] selects HDLC mode, that UCC functions as an HDLC controller. When a UCC in HDLC mode is used with a non multiplexed modem interface, the UCC 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 UCCs. The user can allocate external buffer descriptors (BDs) for receive and transmit tasks so many frames can be sent or received without core intervention.

27.1.1 Block Diagram

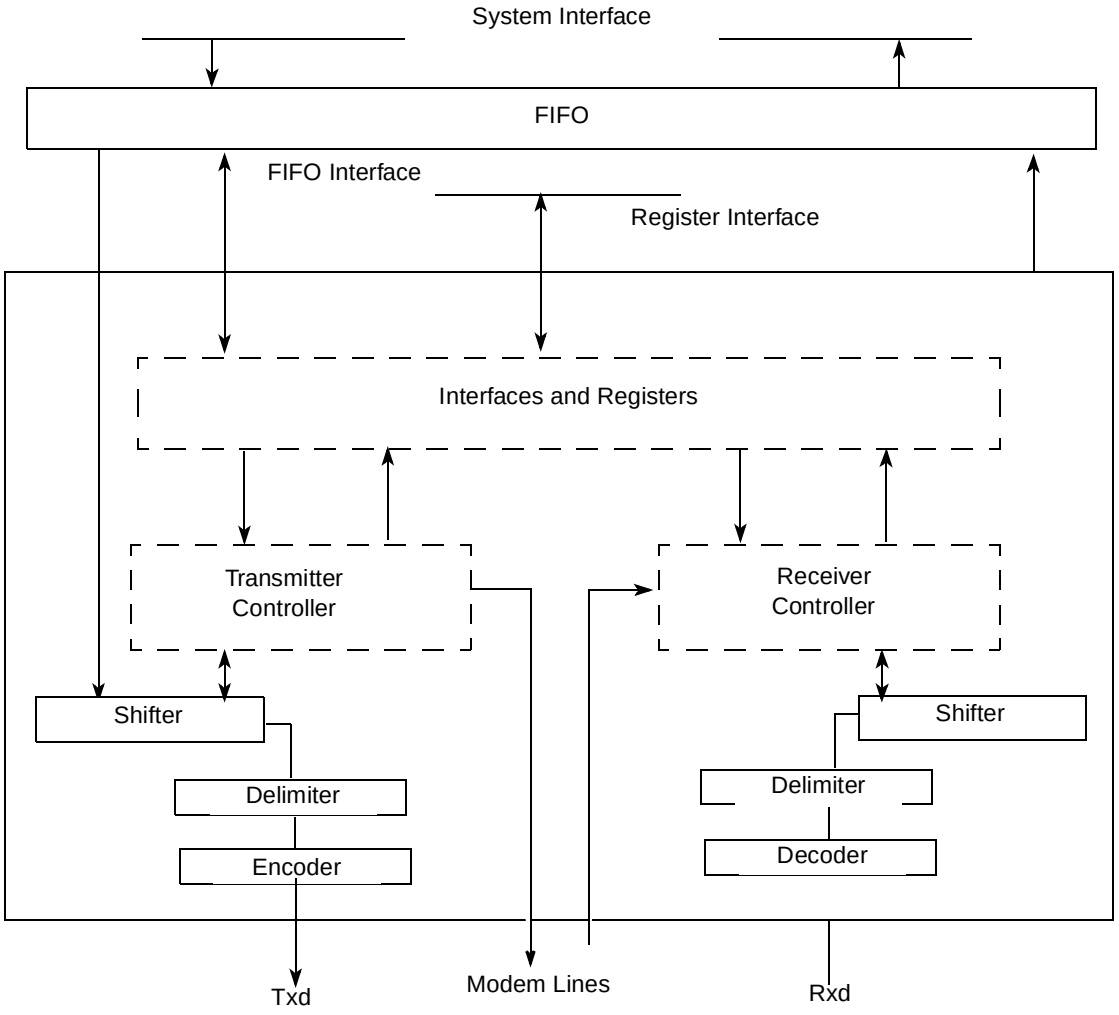


Figure 27-1. HDLC Block Diagram

27.1.2 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 70 Mbps rate
- Support of time stamp mode for Rx frames
- Support of nibble mode HDLC (4 bits per clock)
- Automatic retransmission in case of collision (HDLC bus).

27.1.3 Modes of Operation

The HDLC module can work with several modes of operation which are determined in the UPSMR register.

1. Normal operation (1 bit of data per clock)
2. Nibble mode (4 bits data per clock)
3. HDLC bus 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 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 standards and cannot replace devices that implement these protocols. Instead, it is more suited to non-ISDN LAN and point-to-multipoint configurations.

27.2 Memory Map/Register Definition

27.2.1 Overview

Table 27-1. UCC HDLC Register Summary

Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x4	UPSMR HDLC Mode Register	R/W	0x0000_0000	27.2.2.2/27-7
0x10	UCCE UCC HDLC Event Register	R/W	0x0000_0000	27.2.2.5/27-12
0x14	UCCM UCC HDLC Mask Register	R/W	0x0000_0000	27.2.2.5/27-12
0x18	UCCS UCC HDLC Status Register	R/W	0x0000_0000	27.2.2.6/27-15

27.2.2 Register Descriptions

27.2.2.1 HDLC Parameter RAM

When a UCC operates in HDLC mode, the protocol area of the UCC parameter RAM is mapped with the HDLC parameters as shown in Table 27-2.

Table 27-2. HDLC Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x00	**RIPTR**	Hword	Receive internal temporary data pointer. Used by microcode as a temporary buffer for data. 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. 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 UCC 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 UCC 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 UCC is operating. Change MRBLR only when the UCC receiver is disabled.
0x08	**RSTATE**	Word	Receive internal state. The high byte, RSTATE[0–7], contains the Bus Mode Register, see Section 26.4.6, “Bus Mode Registers (RBMR and TBMR).” RSTATE[8–31] is used by the QUICC Engine module 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 UCC RxBDs. This provides great flexibility in how UCC RxBDs are partitioned. By selecting RBASE entries for all UCCs 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 UCC. The user must initialize RBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled UCCs to overlap or erratic operation occurs.
0x10	RBDSTAT	Hword	RxBD status and control. Reserved for QUICC Engine module use only.
0x12	RBDLEN	Hword	RxBD data length. A down-count value initialized by the QUICC Engine module 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 Bus Mode Register, see Section 26.4.6, “Bus Mode Registers (RBMR and TBMR).” TSTATE[8–31] is used by the QUICC Engine module and must be cleared initially.
0x1C	**TBASE**	Word	TxBD base address (must be divisible by eight). Defines the starting location in the memory map for the UCC TxBDs. This provides great flexibility in how UCC TxBDs are partitioned. By selecting TBASE entries for all UCCs 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 UCC. The user must initialize TBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled UCCs to overlap or erratic operation occurs.

Table 27-2. HDLC Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
0x20	TBDSTAT	Hword	TxBD status and control. Reserved for QUICC Engine module 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 QUICC Engine module 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 QUICC Engine module 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	—	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**[2]	Hword	Discard frame counter. Counts error-free frames discarded due to lack of buffers.
0x4E	**CRCEC**[2]	Hword	CRC error counter. Counts frames not addressed to the user or frames received in the BSY condition, but does not include overrun, $\overline{\text{CD}}$ lost, or abort errors.
0x50	**ABTSC**[2]	Hword	Abort sequence counter
0x52	**NMARC**[2]	Hword	Non-matching address Rx counter. Counts non-matching 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.

Table 27-2. HDLC Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
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 non-matching 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 27-2.
0x60	**HADDR1**	Hword	
0x62	**HADDR2**	Hword	
0x64	**HADDR3**	Hword	
0x66	**HADDR4**	Hword	
0x68	TS_TMP	Hword	Temporary storage
0x6A	TMP_MB	Hword	Temporary storage
0x100	—	—	Reserved

1 Offset from UCC page base address.

2 DISFC, CRCEC, ABTSC, and NMARC—These 16-bit (modulo 216) counters are maintained by the RISC. The user can initialize them while the channel is disabled.

Figure 27-2 shows an example of using HMASK and HADDR[1–4].

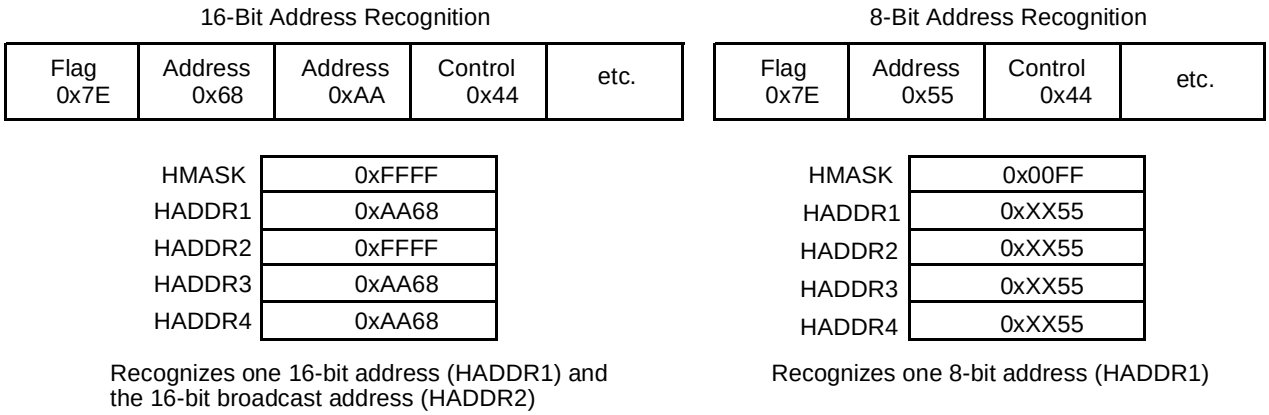


Figure 27-2. HDLC Address Recognition Example

27.2.2.2 HDLC Mode Register (UPSMR)

When a UCC is configured for HDLC mode, the UPSMR is used as the HDLC mode register, shown in Figure 27-3.

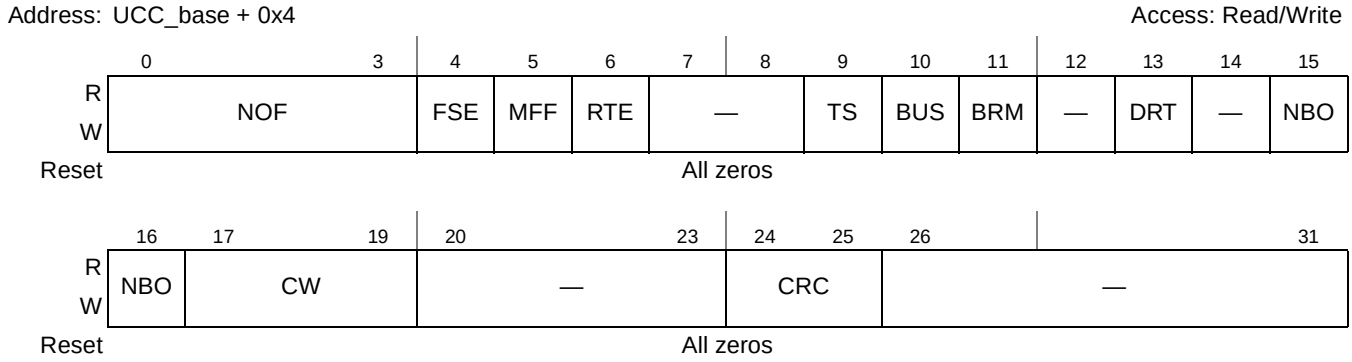


Figure 27-3. HDLC Mode Register (UPSMR)

The UPSMR fields are described in Table 27-3.

Table 27-3. UPSMR Field Descriptions

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 GUMR[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. When UPSMR[BUS] is set, MFF must be cleared. 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.
6	RTE	Retransmit enable. When UPSMR[BUS] is set, RTE must be set. 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.
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 19.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)." Note that when UPSMR[TS] is set, the timestamp used is CETSCR2.

Table 27-3. UPSMR Field Descriptions (continued)

Bits	Name	Description
10	BUS	HDLC bus mode. 0 Normal HDLC operation. 1 HDLC bus operation is selected. See Section 27.3.3, “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	—	Reserved, should be cleared.
13	DRT	Disable receiver while transmitting. 0 Normal operation. 1 As the UCC 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 UCC does not need to receive its own transmission. **Note:** If DRT is set and $\overline{\text{CTS}}$ is not connected to the external device, it is required to programmed the $\overline{\text{CTS}}$ PIO to default.
14	—	Reserved, should be cleared.
15–16	NBO	Mode of operation 00 normal mode (1 bit of data per clock). 01 nibble mode (4 bits of data per clock). 10 octal mode (8 bits of data per clock). 11 Reserved
17–19	CW	Collision window. Valid only if BUS = 1. The number of bytes (0-7) from the beginning of the frame, that a collision could occur. When CW = 000, the collision window is 1 byte. When CW = 111, the collision window is 8 bytes.
20–23	—	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.

27.2.2.3 HDLC Receive Buffer Descriptor (RxBD)

The HDLC controller uses the RxBD to report on data received for each buffer. Figure 27-4 shows an example of the RxBD process.

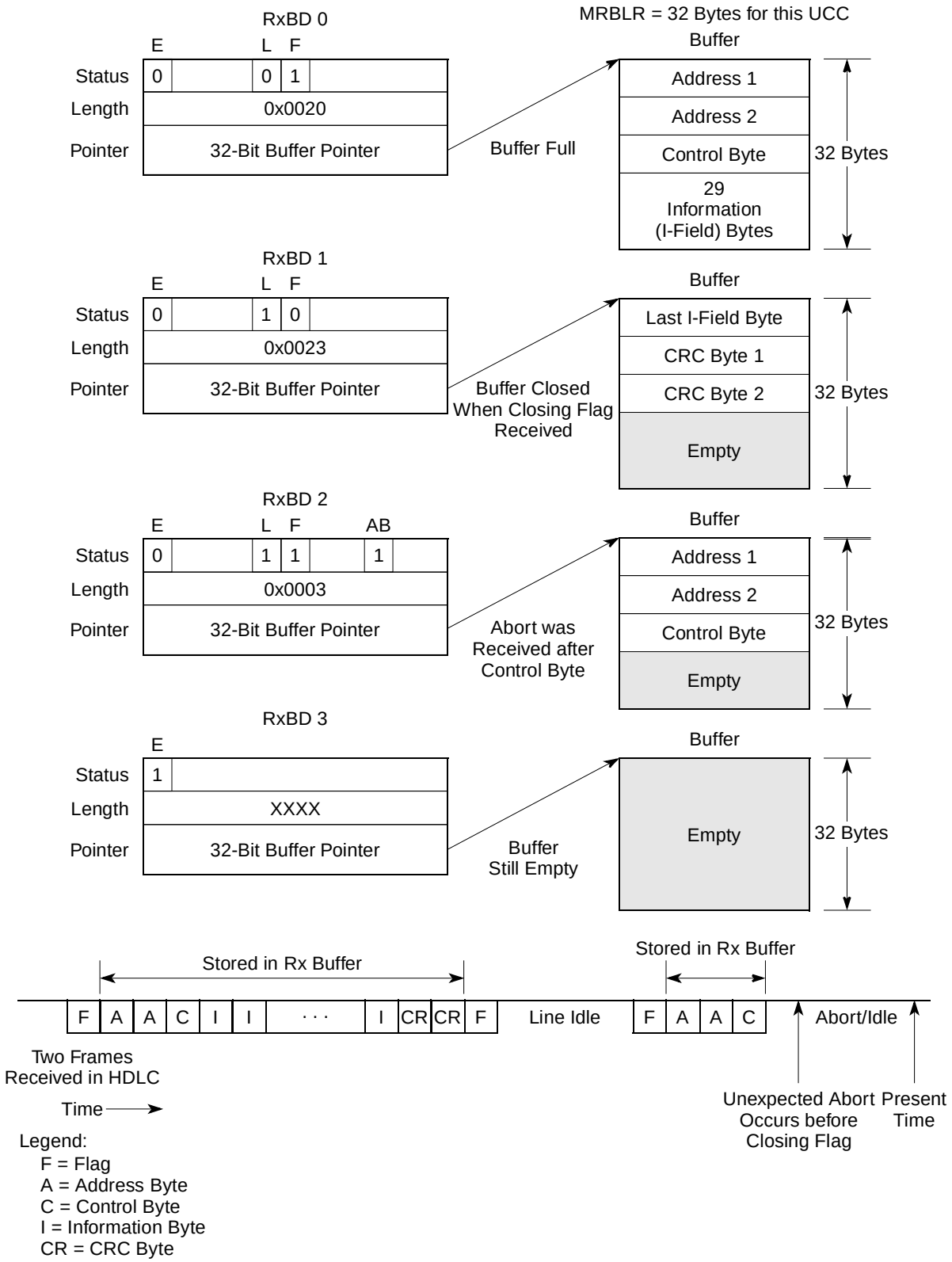


Figure 27-4. UCC HDLC Receiving Using RxBDs

Figure 27-5 shows the UCC HDLC RxBD.

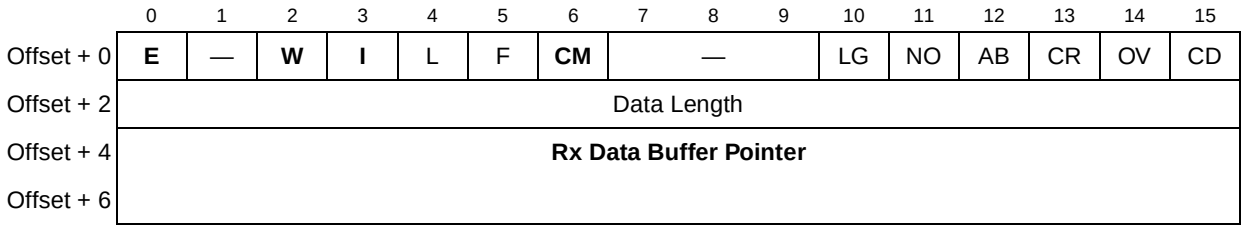


Figure 27-5. UCC HDLC Receive Buffer Descriptor (RxBD)

Table 27-4 describes RxBD fields.

Table 27-4. UCC HDLC RxBD Field Descriptions

Bits	Name	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 RISC 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 RISC. 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 RISC 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 multi-user 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 UCCE[RXB] or UCCE[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 RISC after this BD is closed, allowing the associated data buffer to be automatically overwritten the next time the RISC 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 27-4. UCC HDLC RxBD Field Descriptions (continued)

Bits	Name	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.

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 RISC writes into this BD's data buffer. It is written by the RISC 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 UPSMR[TS] = 1 (see Table 27-3).

27.2.2.4 HDLC Transmit Buffer Descriptor (TxBD)

Data is presented to the HDLC controller for transmission on a UCC 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 27-6 shows the UCC HDLC TxBD.

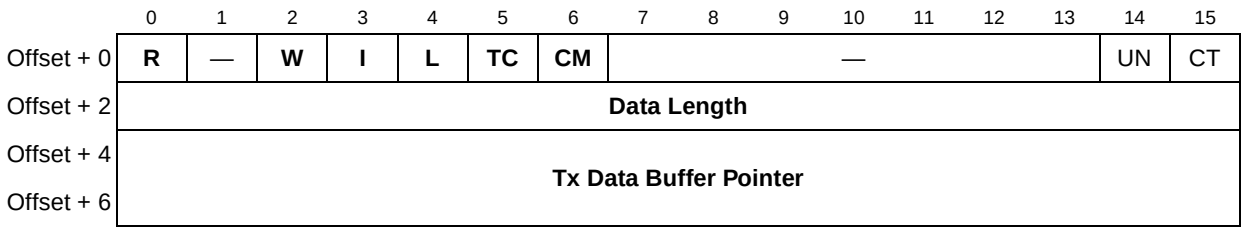


Figure 27-6. UCC HDLC Transmit Buffer Descriptor (TxBD)

Table 27-5 describes HDLC TxBD fields.

Table 27-5. UCC HDLC TxBD Field Descriptions

Bits	Name	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 RISC 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.

Table 27-5. UCC HDLC TxBD (continued) Field Descriptions

Bits	Name	Description
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 RISC 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 multi-user RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 Either UCCE[TXB] or UCCE[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 RISC after this BD is closed, allowing the buffer to be retransmitted automatically the next time the RISC 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.

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 RISC. 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 RISC.

27.2.2.5 HDLC Event Register (UCCE)/Mask Register (UCCM)

The UCCE is used as the HDLC event register when the UCC operates as an HDLC controller. The UCCE reports events recognized by the HDLC channel and generates interrupts. On recognition of an event, the HDLC controller sets the corresponding UCCE bit. UCCE bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the RISC clears the internal interrupt request.

Interrupts generated by the UCCE can be masked in the HDLC mask register (UCCM), which has the same bit format as UCCE. If a UCCM bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked.

Figure 27-7 represents the UCCE/UCCM.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM)

Access: User Read/Write

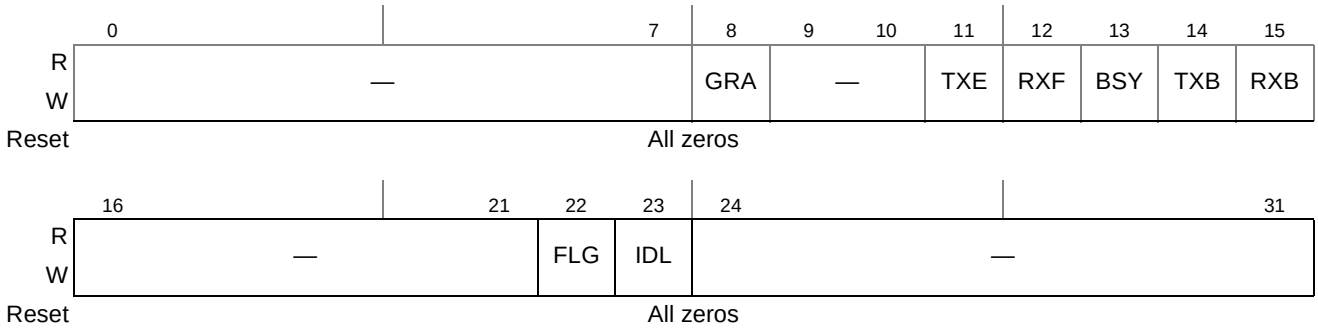


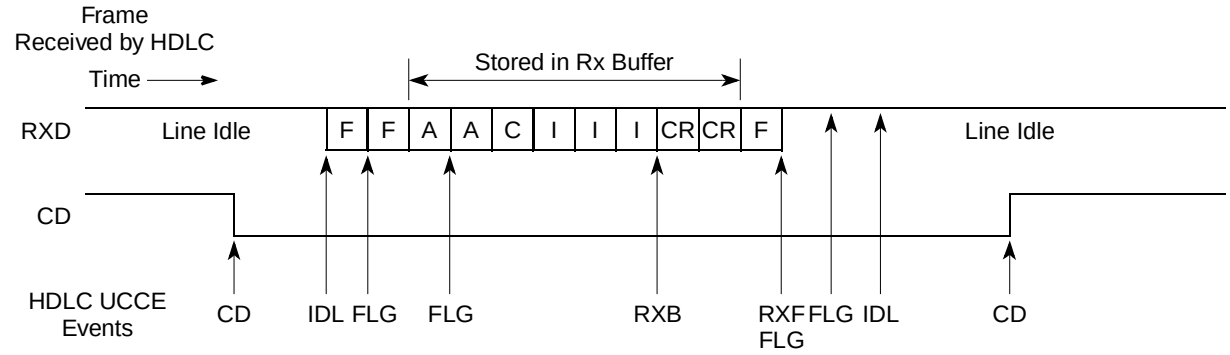
Figure 27-7. HDLC Event/Mask Register (UCCE/UCCM)

Table 27-6 describes UCCE/UCCM fields.

Table 27-6. UCCE/UCCM Field Descriptions

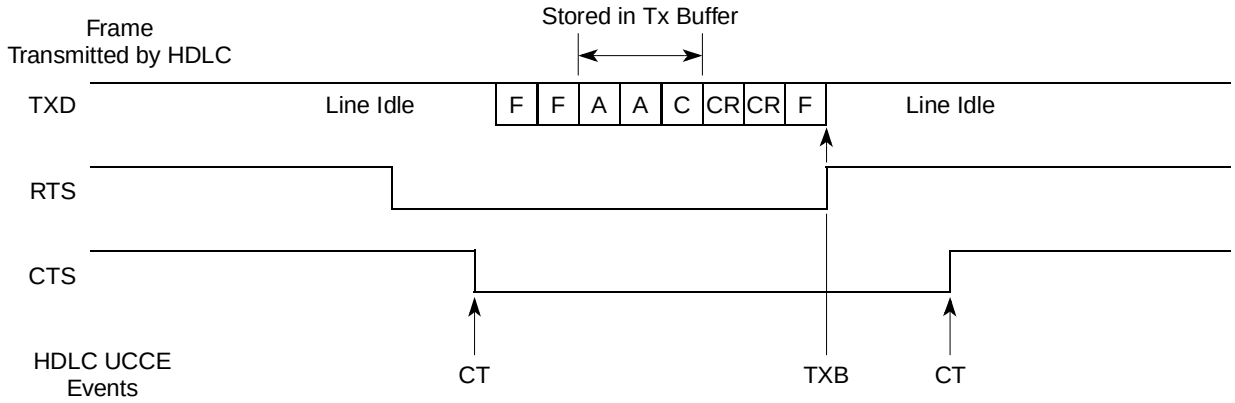
Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. GRA is set as soon as the transmitter finishes transmitting any frame that is in progress when the command was issued. It 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.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when a buffer is filled, even if the buffer is the last in a frame (unlike RXB in the UCCE/UCCM).
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 UCCS; see Section 27.2.2.6, “UCC Status Register (UCCS).”
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 UCCS; see Section 27.2.2.6, “UCC Status Register (UCCS).”
24–31	—	Reserved, should be cleared.

Figure 27-8 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 UCC 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 UCC itself.

Figure 27-8. HDLC Interrupt Event Example

27.2.2.6 UCC Status Register (UCCS)

The UCCS register, shown in Figure 27-9, allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{CTS}$ and $\overline{CD}$ signals are part of the parallel I/O port.

Address: UCC_base + 0x18 Access: Read Only

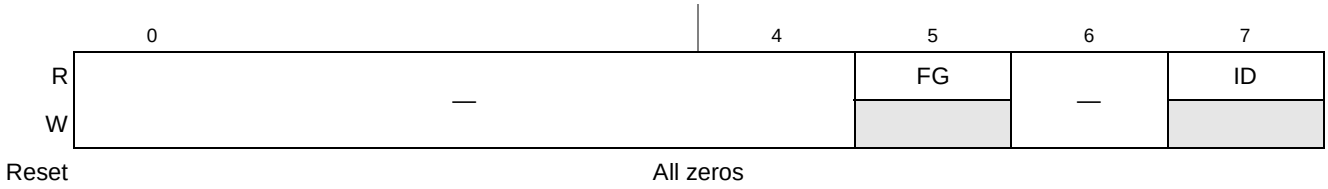


Figure 27-9. HDLC Status Register (UCCS)

Table 27-7. HDLC Status 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.

27.3 Functional Description

27.3.1 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 (UPSMR). 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 UCC 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 27-10 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 27-10. 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 multi buffer 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 RISC 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.

27.3.2 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, and if empty, begins 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, 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.

27.3.3 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 one stops transmitting. The station that sent a zero 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.

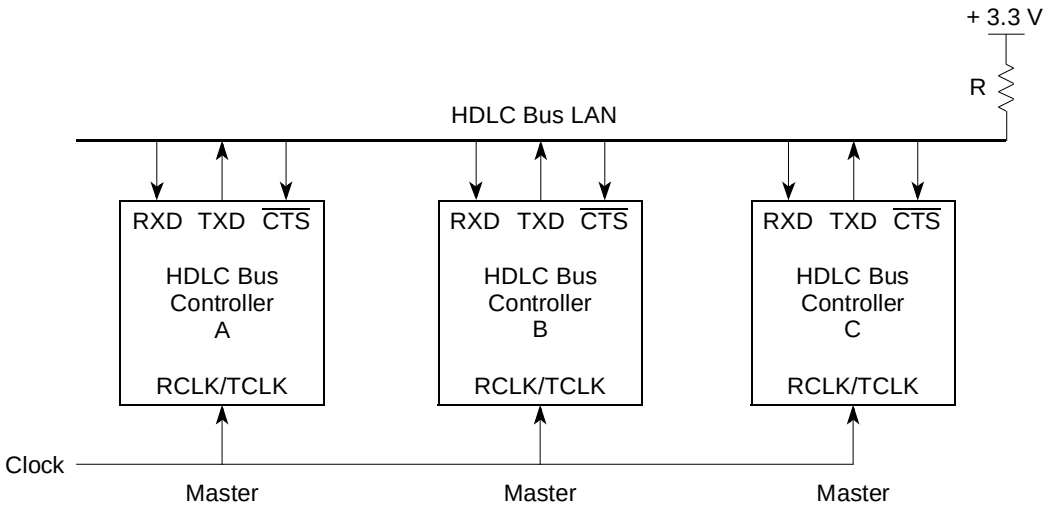
The HDLC bus differs from the I.430 and T1.605 standards as follows:

- 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
- Open-drain connection through port pin configuration or through external transceivers

Figure 27-11 shows the most common HDLC bus LAN configuration, a multi master configuration. A station can transfer data to or from any other LAN station. Transmissions are half-duplex, which is typical in LANs.

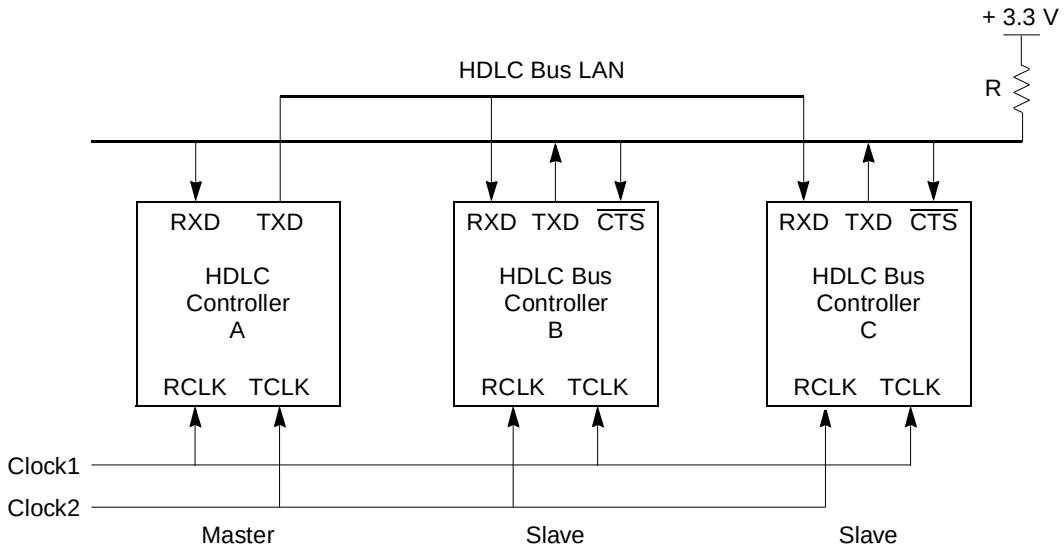


NOTES:

1. Transceivers can 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 27-11. Typical HDLC Bus Multi 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 27-12 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 27-12. Typical HDLC Bus Single-Master Configuration

27.3.3.1 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

27.3.3.2 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. Because the HDLC bus uses a wired-OR scheme, a transmitted zero has priority over a transmitted 1. Figure 27-13 shows how $\overline{\text{CTS}}$ is used to detect collisions.

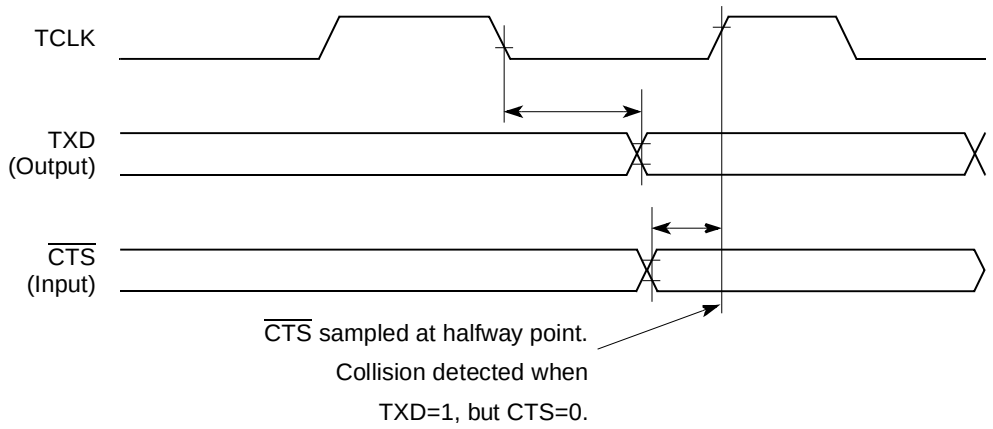


Figure 27-13. Detecting an HDLC Bus Collision

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.

27.3.3.3 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 27-14.

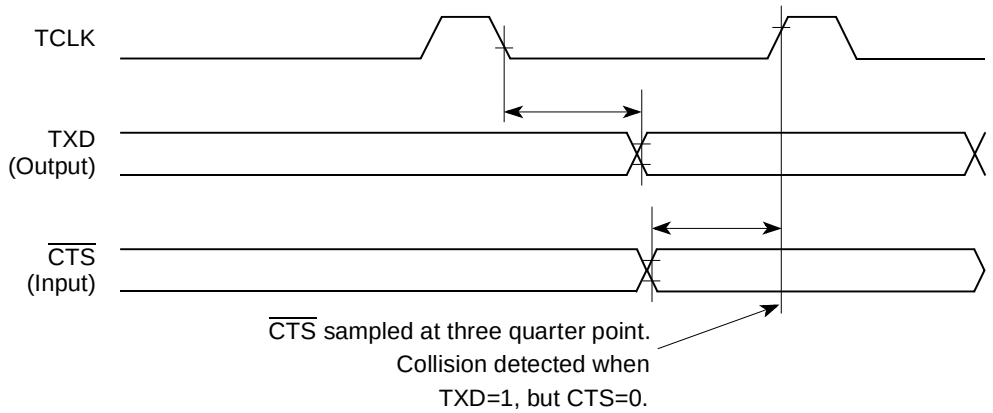
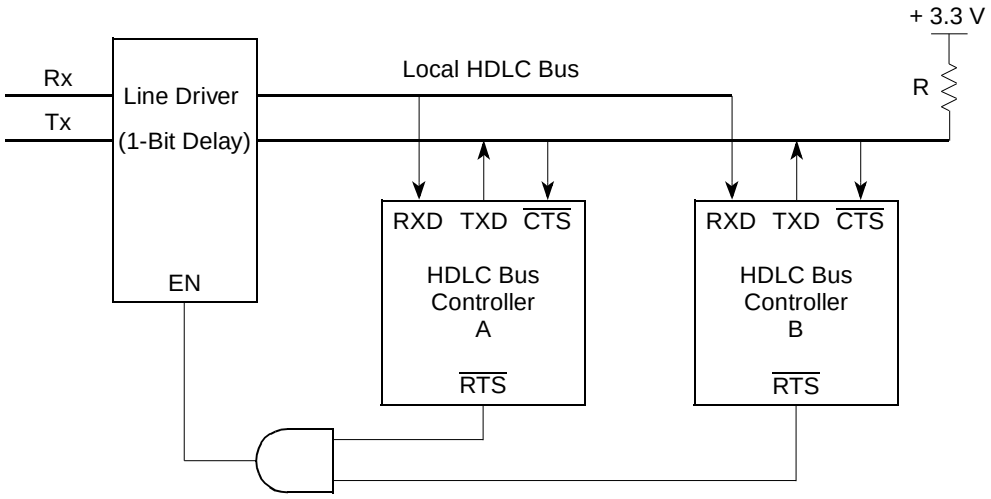


Figure 27-14. Non Symmetrical Tx Clock Duty Cycle for Increased Performance

27.3.3.4 Delayed $\overline{\text{RTS}}$ Mode

Figure 27-15 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 RTS pins of each HDLC bus controller are configured to delayed RTS mode.

Figure 27-15. HDLC Bus Transmission Line Configuration

Normally, $\overline{\text{RTS}}$ goes active at the beginning of the opening flag's first bit. Setting UPSMR[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 27-16 shows $\overline{\text{RTS}}$ timing.

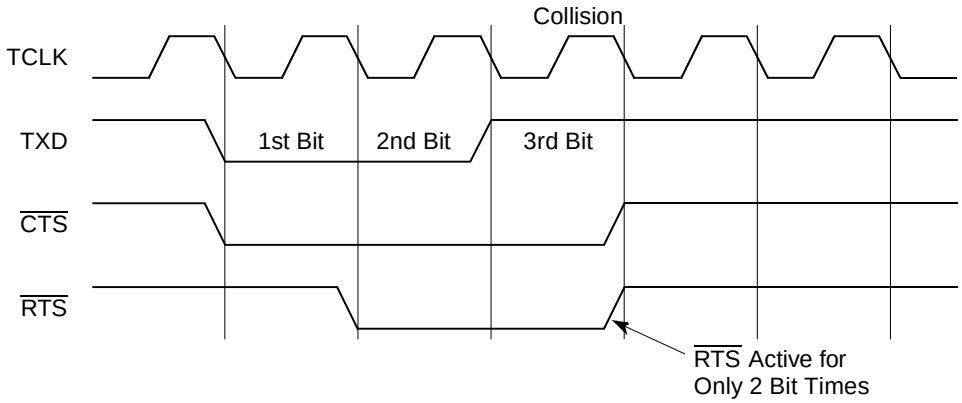
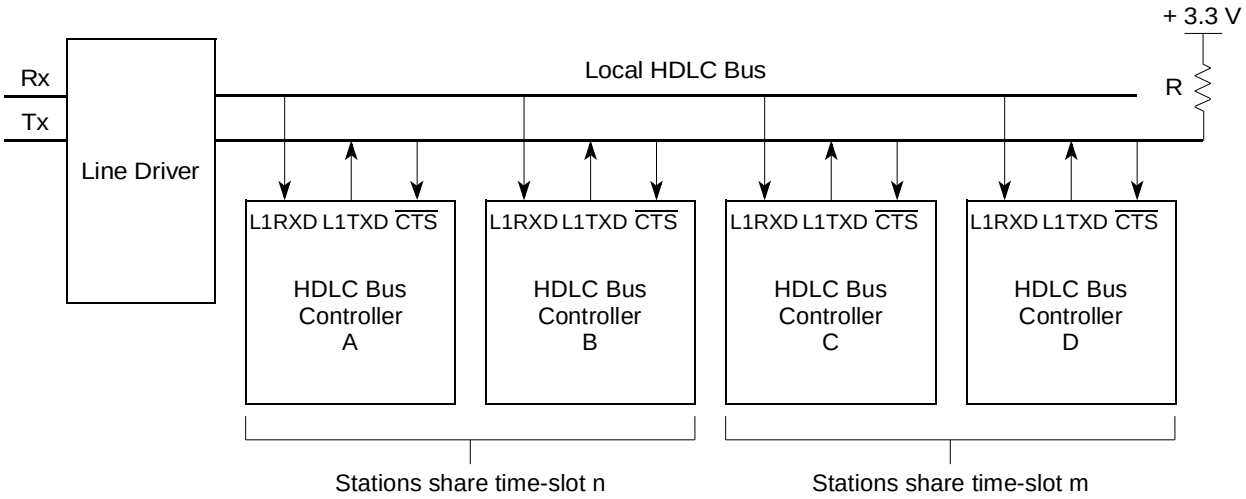


Figure 27-16. Delayed $\overline{\text{RTS}}$ Mode

27.3.3.5 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 27-17. 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 27-17. HDLC Bus TDM Transmission Line Configuration

The local UCCs in HDLC bus mode communicate only with the transmission line and not with each other. The UCCs 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 UCC $\overline{\text{CTS}}$ pin, it must be configured to connect to the chosen UCC. Because the UCC only receives clocks during its time slot, $\overline{\text{CTS}}$ is sampled only during the Tx clock edges of the particular UCC time slot.

27.3.3.6 HDLC Command Set

The transmit and receive commands are issued to the CECR; see Section 19.3.1, “QUICC Engine Command Register (CECR).”

Table 27-8 describes the transmit commands that apply to the HDLC controller.

Table 27-8. Transmit Commands

Command	Description
STOP TRANSMIT	After the hardware or software is reset and the channel is enabled in the UCC 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). The 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 UPSMR[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. UCCE[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.
RESTART TRANSMIT	Enables character transmission on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command is issued and the channel in its UCC 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 27-9 describes the receive commands that apply to the HDLC controller.

Table 27-9. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel is enabled in the UCC 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.

27.3.3.7 HDLC Error Handling

The HDLC controller reports frame reception and transmission error conditions using the channel BDs, error counters, and HDLC event register (UCCE). Table 27-10 describes HDLC transmission errors, which are reported through the TxBD.

Table 27-10. HDLC Transmission Errors

Error	Description
Transmitter Underrun	When this error occurs, the channel terminates buffer transmission, 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 27-11 describes HDLC reception errors, which are reported through the RxBD.

Table 27-11. HDLC Reception Errors

Error	Description
Overrun Error	The HDLC controller maintains an internal FIFO buffer for receiving data. The RISC 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. When $\overline{\text{CD}}$ is lost during flags or during the first data byte, data does not transfer to the Rx FIFO.
$\overline{\text{CD}}$ Lost after frame reception	For GUMR[CDS]=1, if $\overline{\text{CD}}$ is deasserted 1 to 7 bit time after the closing flag, the channel report this as a new frame $\overline{\text{CD}}$ lost error. It close the RxBD, sets RxBD[CD], RxBD[Length]=1 and generates the RXF interrupt if it is enabled. At this point, the receiver enters hunt mode. In this mode, if CD do not envelope the frame, the line should go idle for 8 bit time or more before $\overline{\text{CD}}$ is deasserted to prevent this error. For GUMR[CDS]=0; if $\overline{\text{CD}}$ is deasserted 0 to 7 bit time after the closing flag the channel report this as a new frame $\overline{\text{CD}}$ lost error. In this mode, the line should go idle for 8 bit time or more before $\overline{\text{CD}}$ is deasserted to prevent this 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.

Table 27-11. HDLC Reception Errors (continued)

<table>
<tr><th>Error</th><th>Description</th></tr>
<tr><td>Nonoctet Aligned Frame</td><td>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:
<table>
<tr><td>msb</td><td></td><td></td><td></td><td></td><td></td><td></td><td>lsb</td></tr>
<tr><td colspan="4">Valid data</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
</table></td></tr>
<tr><td>CRC Error</td><td>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.</td></tr>
</table>

27.4 Initialization Information

27.4.1 HDLC Bus Protocol Programming

The HDLC bus on the QUICC Engine module is implemented using the UCC in HDLC mode with bus-specific options selected in the UPSMR and GUMR, as outlined below.

27.4.1.1 Programming GUMR and UPSMR for the HDLC Bus Protocol

To program the protocol-specific mode register (UPSMR), 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 UCC mode register (GUMR), 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 TENC and RENC for NRZ (0b000).
- Clear RTSM to send idles between frames.
- Set GUMR _L[ENT, ENR] as the last step to begin operation.

Chapter 28
Transparent Controller

The UCC transparent controller, shown in Figure 28-1, functions as a high-speed serial-to-parallel and parallel-to-serial converter. Transparent mode provides a clear channel on which the UCC performs no bit-level manipulation; implementing higher-level protocols requires software. Transparent mode is also referred to as a totally transparent or promiscuous operation.

Basic applications for a UCC in transparent mode include the following:

- Moving data serially, such as voice, 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

A UCC transmitter and receiver can be programmed in transparent mode independently. Setting GUMR*x*[TTx] enables the transparent transmitter; setting GUMR*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 UCC operates with the protocol programmed in GUMR*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. Transparent controllers are the only ones that can be split in this way.

The UCC 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 UCC 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 UCCs. Each clock can be supplied from the internal BRG bank or provided by external signals.

28.1 Block Diagram

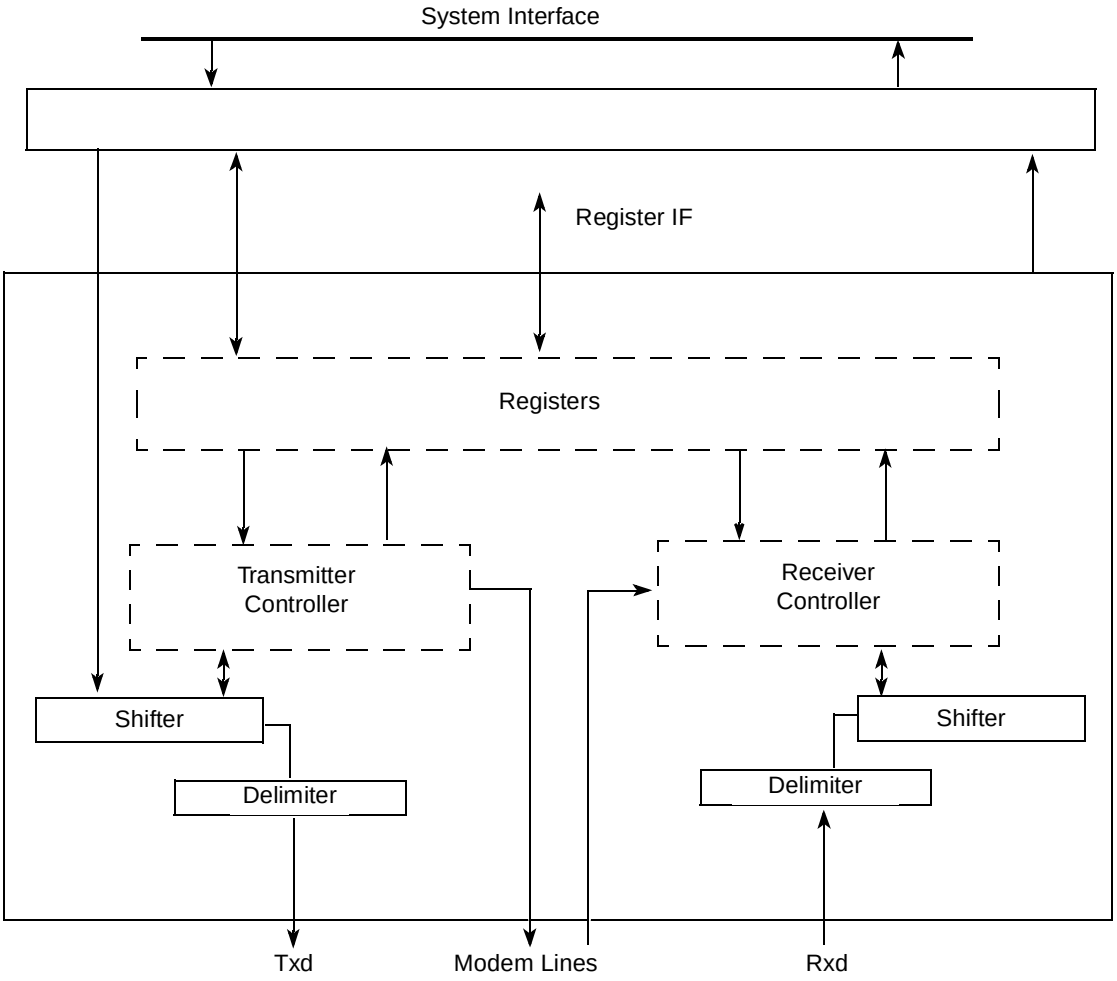


Figure 28-1. Transparent Block Diagram

28.2 Features

The main features of the transparent controller are as follows:

- 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
- Supports transparent nibble/octal modes (4/8 bits per clocks)
- Reverse data mode

- Another protocol can be performed on the UCC's other half (transmitter or receiver) during transparent mode
- External BD table

28.3 Modes of Operation

The transparent protocol modes of operation are determined by the GUMR register and UPSMR register (1 bit/nibble/octal mode).

28.3.1 Carrier Detection Pulse or Normal mode

Normal operation (GUMR[CDP] = 0)

- The $\overline{CD}$ (carrier detection) signal should envelope the frame. Negation of $\overline{CD}$ during frame receiving causes a receiver error ($\overline{CD}$ lost). When this error occurs the receiver will ignore the remaining current frame and wait for the next valid frame arrival. The receiver will set the RxBD[CD] bit.

Pulse mode (GUMR[CDP] = 1)

- Once $\overline{CD}$ is asserted, synchronization has been achieved. Further transitions of $\overline{CD}$ do not affect reception.

28.3.2 Reverse Data Mode

Normal Operation (GUMR[REVD] = 0)

Reverse data (GUMR[REVD] = 1)

- Reverse data modes supported are 1 bit and nibble.
  - **1 Bit mode**: The totally transparent channels on this UCC reverse the bit order, transmitting the msb of each octet first. This can be programmed to happen on the receiver, transmitter, or both, as defined by TTX and TRX.
  - **Nibble mode**: The totally transparent channels on this UCC reverse the nibble order, transmitting the 4 msb's of each octet first. This can be programmed to happen on the receiver, transmitter, or both, as defined by TTX and TRX.

28.3.3 Synchronization Pattern Modes

Determines the operation of a UCC receiver configured for totally transparent operation only.

- GUMR[SYNL] = 00

  The sync pattern in the UDSR is not used. An external sync signal is used instead ($\overline{CD}$ signal asserted: high to low transition).
- GUMR[SYNL] = 01

  Automatic sync (assumes always synchronized, ignores $\overline{CD}$ signal).

  GUMR[SYNL] = 10

8-bit sync. The receiver synchronizes on an 8-bit sync pattern stored in the UDSR. Negation of $\overline{CD}$ causes $\overline{CD}$ lost error.

- GUMR[SYNL] = 11

  16-bit sync. The receiver synchronizes on a 16-bit sync pattern stored in the UDSR. Negation of $\overline{CD}$ causes $\overline{CD}$ lost error.

28.4 Memory Map/Register Definition

28.4.1 Overview

Table 28-1. UCC Transparent Register Summary

Offset[1]	Register	Access	Reset Value	Section/Page
0x0	General UCC Mode Register (GUMR)	R/W		26.4.2.1/26-6
0x4	UCC Transparent Mode Register (UPSMR)	R/W	0	28.4.2.2/28-6
0x8	Transmit On Demand Register (UTODR)	R/W		22.3.4/22-6
0xC	UCC Data Synchronization Register (UDSR)	R/W		26.4.5/26-11
0x10	UCC Transparent Event Register (UCCE)[2]	R/W	0	28.4.2.5/28-10
0x14	UCC Transparent Mask Register (UCCM)	R/W	0	28.4.2.5/28-10

1 From UCC Base.

2 Set by QUICC Engine module, cleared by host.

28.4.2 Register Descriptions

28.4.2.1 UCC Transparent Parameter RAM

When a UCC operates in transparent mode, the protocol area of the UCC parameter RAM is mapped with the transparent parameters in Table 28-2.

Table 28-2. UCC Transparent Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x00	**RIPTR**	Hword	Receive internal temporary data pointer. Used by microcode as a temporary buffer for data. 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. 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.

Table 28-2. UCC Transparent Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
0x06	**MRBLR**	Hword	Maximum receive buffer length (a multiple of 32 for all modes). The number of bytes that the UCC 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 UCC 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 UCC is operating. Change MRBLR only when the UCC receiver is disabled.
0x08	**RSTATE**	Word	Receive internal state. The high byte, RSTATE[0–7], contains the bus mode register, see Section 26.4.6, “Bus Mode Registers (RBMR and TBMR).” RSTATE[8–31] is used by the QUICC Engine module 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 UCC RxBDs. This provides great flexibility in how UCC RxBDs are partitioned. By selecting RBASE entries for all UCCs 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 UCC. The user must initialize RBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled UCCs to overlap or erratic operation occurs.
0x10	RBDSTAT	Hword	RxBD status and control. Reserved for QUICC Engine module use only.
0x12	RBDLEN	Hword	RxBD data length. A down-count value initialized by the QUICC Engine module 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 bus mode register, see Section 26.4.6, “Bus Mode Registers (RBMR and TBMR).” TSTATE[8–31] is used by the QUICC Engine module and must be cleared initially.
0x1C	**TBASE**	Word	TxBD base address (must be divisible by eight). Defines the starting location in the memory map for the UCC TxBDs. This provides great flexibility in how UCC TxBDs are partitioned. By selecting TBASE entries for all UCCs 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 UCC. The user must initialize TBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled UCCs to overlap or erratic operation occurs.
0x20	TBDSTAT	Hword	TxBD status and control. Reserved for QUICC Engine module 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 QUICC Engine module 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.

Table 28-2. UCC Transparent Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
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 QUICC Engine module 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	—	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**[2]	Hword	Discard frame counter. Counts error-free frames discarded due to lack of buffers.
0x4E	**CRCEC**[2]	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.
0x58	—	4 words	Reserved, should be cleared.
0x68	TS_TMP	Hword	Temporary storage
0x6A	TMP_MB	Hword	Temporary storage

1 From UCC*x* page base address

2 DISFC and CRCEC—These 16-bit (modulo 216) counters are maintained by the QUICC Engine module. The user should initialize them while the channel is disabled.

28.4.2.2 UCC Transparent Mode Register (UPSMR)

UPSMR is shown in Figure 28-2. When a UCC is configured for HDLC and transparent, the UPSMR is used as the HDLC mode register, shown in Section 27.2.2.2, “HDLC Mode Register (UPSMR).”

Figure 28-2. UCC Transparent Mode Register (UPSMR)

The UPSMR fields are described in Table 28-3.

Table 28-3. UPSMR Field Descriptions

Bits	Name	Description
0–14	—	Reserved, should be cleared.
15–16	NBO	Mode of operation 00 normal mode (1 bit of data per clock). 01 nibble mode (4 bits of data per clock). 10 octal mode (8 bits of data per clock). 11 Reserved
17–31	—	Reserved, should be cleared.

28.4.2.3 UCC Transparent Receive Buffer Descriptor (RxBD)

Figure 28-3 shows the UCC Transparent RxBD.

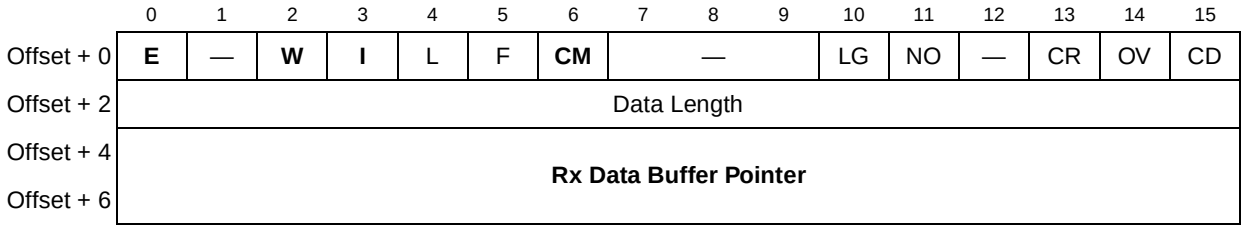


Figure 28-3. UCC Transparent Receive Buffer Descriptor (RxBD)

Table 28-4 describes RxBD fields.

Table 28-4. UCC Transparent RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The buffer is full of 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 multi-user RAM. The RxBD table must contain more than one BD in transparent mode.
3	**I**	Interrupt 0 The RXB bit is not set after this buffer is used, but RXF operation remains unaffected. 1 UCCE[RXB] or UCCE[RXF] is set when the transparent controller uses this buffer. These two bits can cause interrupts if they are enabled.

Table 28-4. UCC Transparent RxBD Field Descriptions (continued)

Bits	Name	Description
4	L	Last in frame. Set by the transparent 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, and/or LG bits are set. The transparent 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 transparent 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.
7–10	—	Reserved, should be cleared.
11	NO	Rx nonoctet-aligned frame. Set when a received frame contains a number of bits not divisible by eight.
12	—	Reserved, should be cleared.
13	CR	Rx CRC error. This frame contains a CRC error. Received CRC bytes are written to the receive buffer.
14	OV	Overrun. Set when 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.

The RxBD status bits are written by the transparent controller after receiving the associated data buffer.

28.4.2.4 Transparent Transmit Buffer Descriptor (TxBD)

Data is presented to the transparent controller for transmission on a UCC channel by arranging it in buffers referenced by the channel TxBD table. The transparent controller confirms transmission (or indicates errors) using the BDs to inform the core that the buffers have been serviced. Figure 28-4 shows the UCC Transparent TxBD.

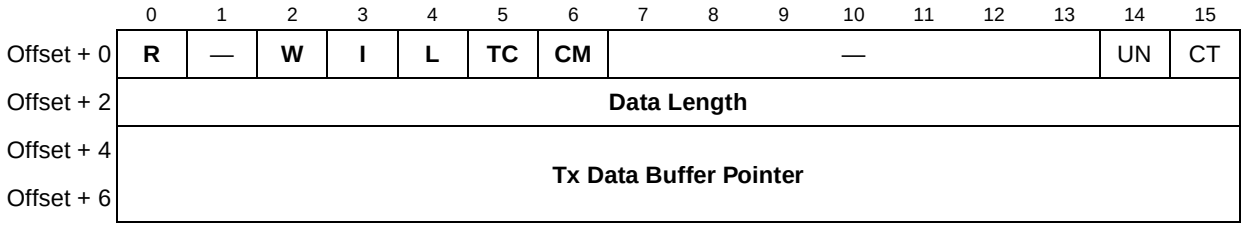


Figure 28-4. UCC Transparent Transmit Buffer Descriptor (TxBD)

Table 28-5 describes Transparent TxBD fields.

Table 28-5. Transparent TxBD Field Descriptions

Bits	Name	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 multi-user RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 Either UCCE[TXB] or UCCE[TXE] is set when this buffer is serviced by the transparent 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 transparent controller encounters a transmitter underrun condition while sending the buffer. The transparent 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 transparent controller writes CT after sending the buffer.

The TxBD status bits are written by the transparent controller after sending the associated data buffer.

The remaining TxBD parameters are as follows:

- Data length is the number of bytes the transparent controller should transmit from this data buffer; it is never modified by the RISC. 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 RISC.

28.4.2.5 UCC Transparent Event Register (UCCE)/Mask Register (UCCM)

UCCE reports events recognized by the transparent channel and generates interrupts. On recognition of an event, the transparent controller sets the corresponding UCCE bit. UCCE bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the RISC clears the internal interrupt request. Interrupts generated by the UCCE can be masked in the UCCM, which has the same bit format as UCCE. If a UCCM bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked. Figure 28-5 represents the UCCE/UCCM.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM) Access: Read/Write

	0–7	8	9–10	11	12	13	14	15
R W	—	GRA	—	TXE	RXF	BSY	TXB	RXB
Reset	All zeros							

Figure 28-5. Transparent Event/Mask Register (UCCE/UCCM)

Table 28-6 describes UCCE/UCCM fields.

Table 28-6. Transparent UCCE/UCCM 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 Transparent 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 Transparent 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.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when a buffer is filled, even if the buffer is the last in a frame.
16–31	—	Reserved, should be cleared.

28.4.2.6 UCC Transparent Commands

The following transmit and receive commands are issued to the RISC command register. Table 28-7 describes the transmit commands.

Table 28-7. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GUMR, 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 functions. 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. UCCE[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.
RESTART TRANSMIT	Re-enables transmission of characters on the transmit channel. The transparent controller expects this after a STOP TRANSMIT command is issued at which point the channel is disabled in UCCM, or 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 the reset state. It should be issued only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

Table 28-8 describes receive commands.

Table 28-8. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel is enabled in the UCC 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 Transparent receiver to abort reception of the current frame and enter the hunt mode. In hunt mode, the Transparent controller continually scans the input data stream for the sync 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 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.

28.4.2.7 Handling Errors in the Transparent Controller

The UCC reports message reception and transmission errors using the channel buffer descriptors, the error counters, and UCC Transparent Event Register (UCCE). Table 28-9 describes transmit errors.

Table 28-9. Transparent Transmission 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 UCCE[TXE] is set. Underrun cannot occur between transparent frames.
$\overline{\text{CTS}}$ Lost During Frame 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 28-10 describes receive errors.

Table 28-10. Transparent Reception Errors

Error	Description
Overrun Error	The transparent controller maintains an internal FIFO buffer for receiving data. The RISC 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. When GUMR[SYNL] = 1x (8-bit sync or 16-bit sync) and $\overline{\text{CD}}$ is lost during syncs or during first data byte, data does not transfer to the Rx FIFO.
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.

28.5 Functional Description

28.5.1 UCC Transparent Channel Frame Transmission Process

The transparent transmitter is designed to work with almost no intervention from the core. When the core enables the UCC transmitter in transparent mode, it starts sending idles, which are logic high or encoded ones, as programmed in GUMR[TEND]. The UCC polls the first BD in the TxBD table. When there is a message to send, the UCC fetches data from memory, loads the transmit FIFO, and waits for transmitter synchronization, which is achieved with $\overline{\text{CTS}}$ before the transmitter begins sending (see Section 28.5.3, "Achieving Synchronization in Transparent Mode"), or by waiting for the receiver to achieve synchronization, depending on GUMR[TXSY]. Transmission begins when transmitter synchronization is achieved.

When all BD data has been sent, if TxBD[L] is set, the UCC 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 UCCE[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 UCC then proceeds to the next BD in the table and any whole number of bytes can be sent. If GUMR[REVD] is set, the bit order of each byte is reversed before being sent, the msb of each octet is sent first.

An optional CRC, selected in GUMR[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.

28.5.2 UCC Transparent Channel Frame Reception Process

When the core enables the UCC 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 UCC 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, UCCE[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 GUMR[REVD] is set, the bit order of each byte is reversed before it is written to memory.

The receiver always checks the CRC of the received frame, according to GUMR[TCRC]. If a CRC is not required, resulting errors can be ignored.

28.5.3 Achieving Synchronization in Transparent Mode

Once the UCC transmitter is enabled for transparent operation in the GUMR, the TxBD is prepared for the UCC, and the transmit FIFO is preloaded by the SDMA channel. Transmit synchronization must be established before data can be sent.

Similarly, once the UCC receiver is enabled for transparent operation in the GUMR and the RxBD is made empty for the UCC, 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

28.5.3.1 Synchronization in NMSI Mode

This section describes synchronization in NMSI mode.

28.5.3.1.1 In-Line Synchronization Pattern

The transparent channel can be programmed to transmit and receive a synchronization pattern if GUMR[SYNL] ≠ 0; see Section 26.4.2.1, “GUMR in Fast Mode.” The pattern is defined in the UDSR; see Section 26.4.5, “UCC Data Synchronization Register (UDSR).” GUMR[SYNL] defines the SYNC pattern length. The synchronization pattern is shown in Figure 28-6.

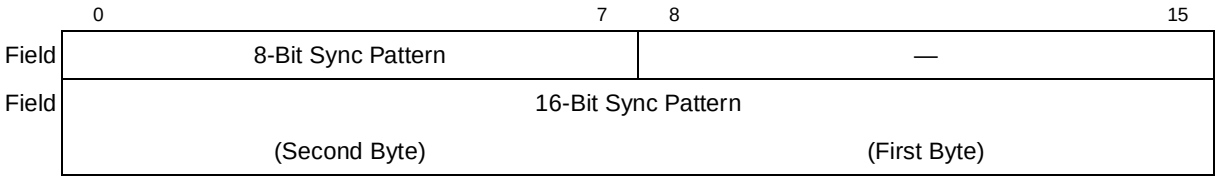


Figure 28-6. In-Line Synchronization Pattern

The receiver synchronizes on the synchronization pattern located in the UDSR. 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.

NOTE

The transparent controller does not automatically send the synchronization pattern; therefore, the synchronization pattern must be included in the transmit buffer.

28.5.3.1.2 External Synchronization Signals

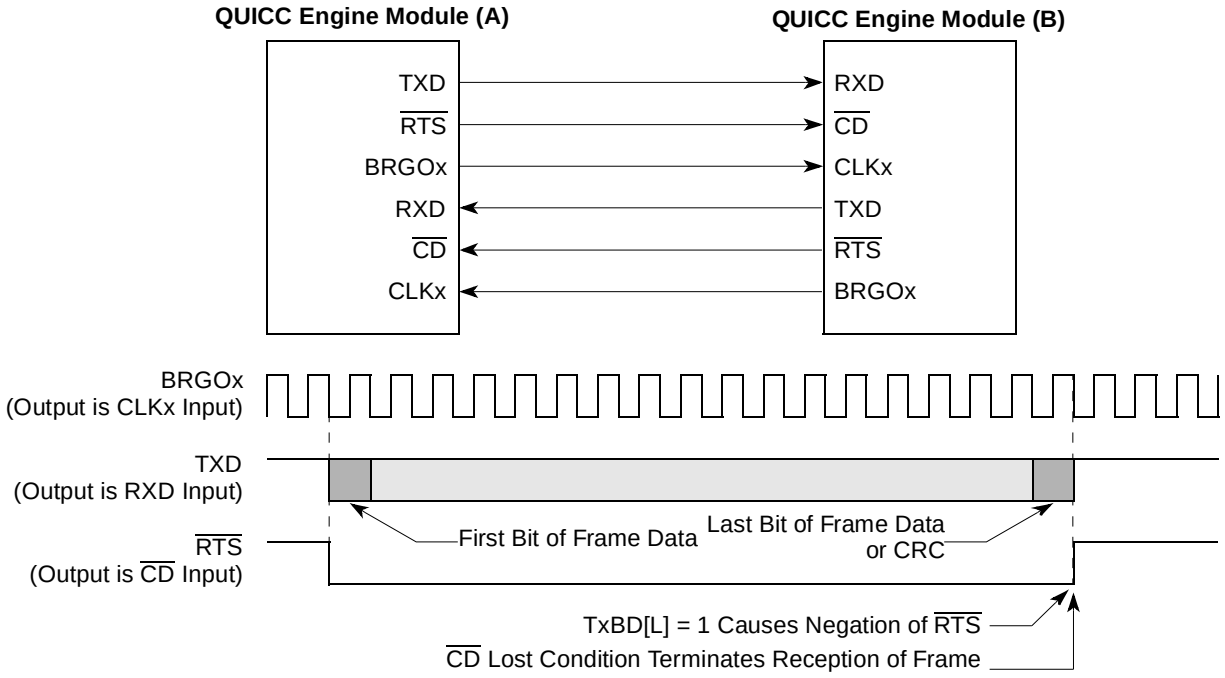
If GUMR[SYNL] = 00, an external signal is used to begin the sequence. $\overline{\text{CTS}}$ is used for the transmitter and $\overline{\text{CD}}$ is used for the receiver; these signals share the following sampling options.

- The pulse/envelope option determines whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ needs 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 GUMR. 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.
- The sampling option determines the delay between $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ being asserted and the resulting action by the UCC. These signals can be assumed to be asynchronous to the data and then internally synchronized by the UCC, or they can be assumed to be synchronous to the data giving faster operation. This option allows the $\overline{\text{RTS}}$ of one UCC to be connected to the $\overline{\text{CD}}$ of another UCC (on another device) 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 in Section 22.5, “Controlling UCC Timing with RTS, CTS, and CD.”

28.5.3.1.3 Transparent Synchronization Example

Figure 28-7 shows an example of synchronization using external signals.



Notes:

1. Each QUICC Engine module generates its own transmit clocks. If the transmit and receive clocks are the same, one can generate transmit and receive clocks for the other QUICC Engine device. For example, CLKx on QUICC Engine module (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 GUMR 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.
5. The transparent frame contains a CRC if TxBD[TC] is set.

Figure 28-7. Sending Transparent Frames Between QUICC Engine Modules or Other QUICC Engine Devices

QUICC Engine module (A) and QUICC Engine module (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 QUICC Engine module's $\overline{\text{CD}}$. GUMR[SYNL] is not used and transmission and reception from each QUICC Engine module are independent.

28.5.3.1.4 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 UCC in loopback mode and then change GUMR[DIAG] to 0b00 while the transmitter and receiver are enabled.

28.5.3.2 Synchronization and the TSA

A 1 bit transparent-mode UCC using the time-slot assigner can synchronize either on a user-defined inline pattern or by inherent synchronization. Note that when the TSA is used, 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.

28.5.3.2.1 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 GUMR[SYNL].
- Program the UDSR with the desired pattern.
- Clear GUMR[CDP].
- Set GUMR[CTSP, CTSS, CDS].

If GUMR[TXSY] is also used, the transmitter begins transmission eight clocks after the receiver achieves synchronization.

28.5.3.2.2 Inherent Synchronization

Inherent synchronization assumes synchronization by default when the channel is enabled; all data sent from the TDM to the UCC is received. To implement inherent synchronization, set GUMR[CDP, CDS, CTSP, CTSS]. If these bits are not set, the received bit stream is bit-shifted. The UCC loses the first received bit because $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are treated as asynchronous signals.

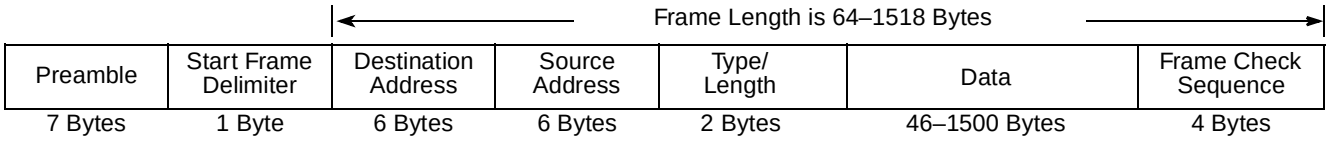
28.5.3.2.3 End of Frame Detection

An end of frame cannot be detected in the transparent data stream because 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.

Chapter 29
UCC Ethernet Controller (UEC)

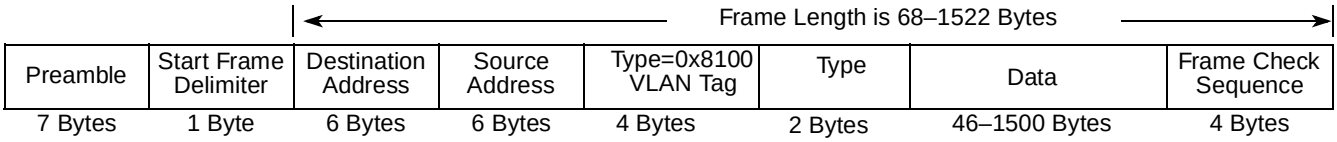
29.1 Introduction

The Ethernet IEEE 802.3 protocol is a widely-used LAN, based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 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 frames are based on the frame structure shown in Figure 29-1.



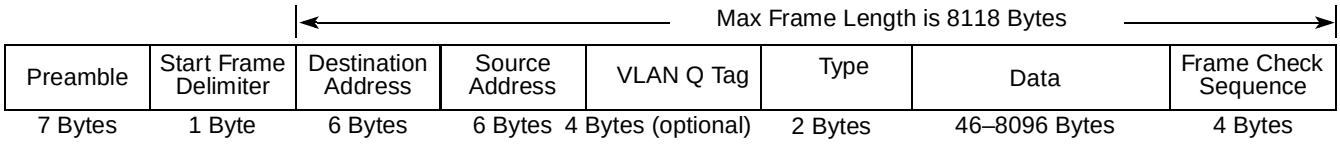
Note: The lsb of each octet is transmitted first.

Figure 29-1. Untagged Ethernet Frame Structure



Note: The lsb of each octet is transmitted first.

Figure 29-2. VLAN Tagged Ethernet Frame Structure



Note: The lsb of each octet is transmitted first.

Figure 29-3. VLAN Tagged Jumbo Ethernet Frame Structure

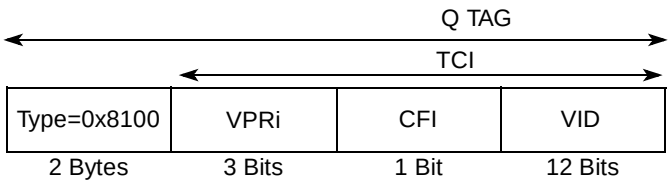


Figure 29-4. VLAN Q TAG

Note that the maximum frame length is user programmable. The figures depict a few examples.

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 specification allowed 16-bit addressing, which has never been widely used.
- Ethernet type field/IEEE 802.3 length field. The UEC considers the two bytes immediately following the MAC source address as the frame length if its numerical value is less or equal to 0x600. Otherwise the field is considered as a type field. Specifically, if a value of 0x8100 is detected, the UEC assumes a VLAN tag is present.
- Data
- 4-byte frame-check sequence (FCS), which is the standard 32-bit CCITT-CRC polynomial used in many protocols.

Figure 29-5 describes the block diagram of the UCC Ethernet controller.

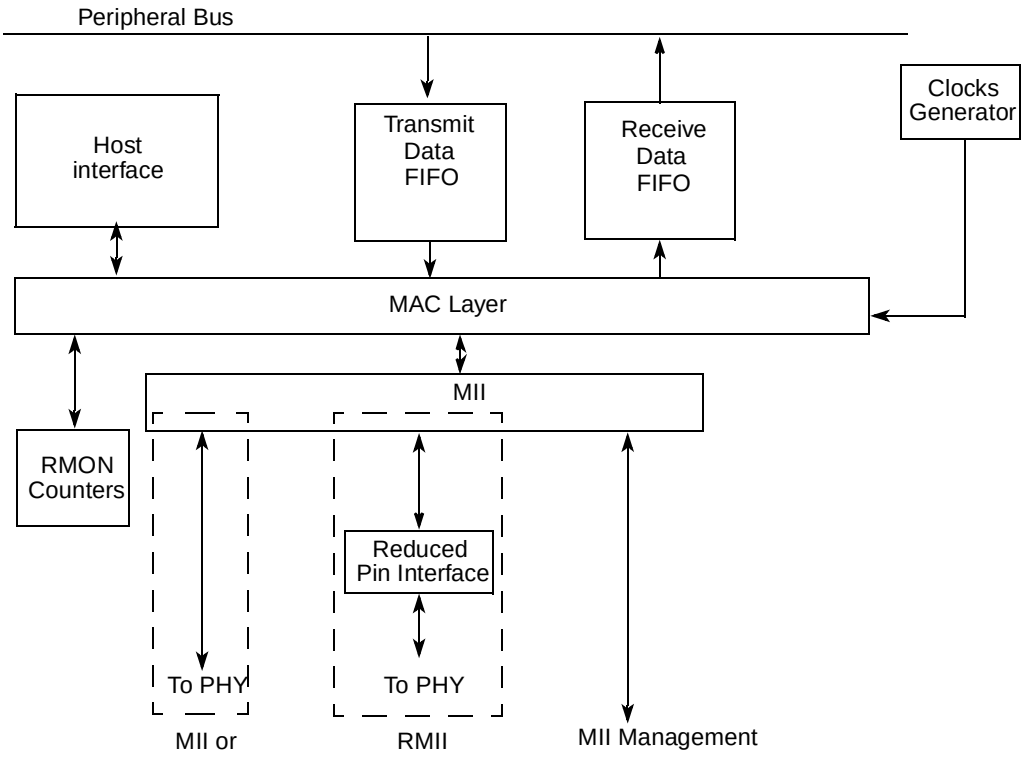


Figure 29-5. UCC Ethernet Controller Block Diagram

29.2 Overview

The UCC Ethernet Controller is a Fast Ethernet Controller.

The UEC, supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver.

- 10/100 Mbps MII interface (IEEE 802.3-2002)

- RMII interface

The MII 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.

The RMII interface (low pin count Reduced Media Independent Interface as specified by the RMII Consortium™ standard), is a reduced MII interface. The purpose of this interface is to provide a low cost alternative to the MII, with an 8-pin interface instead of 16 (MDC and MDIO pins not included).

29.3 Features

The UCC Ethernet Controller includes the following features:

- Support for different Ethernet physical interfaces:
  - 10/100 Mbps IEEE 802.3 MII
  - 10/100 Mbps RMII (Consortium standard)
- Supports all RMII specification features:
  - Capable of supporting 10 and 100 Mbps data rates
  - Supporting half- and full-duplex 10/100Mbps
  - Provides independent 2-bit wide (di-bit) transmit and receive data paths
- Support half-duplex back-pressure (10/100 Mbps only)
  - Backpressure Control via Host Command
  - Automatic backpressure according to programmable Receive Thresholds
- Supports PAUSE Flow Control for full duplex operation
  - Transmit Flow Control via a Host command
  - Automatic Transmit Automatic Flow Control according to programmable Receive FIFO Thresholds
  - Programmable MAC Parameter in Flow Control frame
  - Automatic resend of Flow Control frame if MAC parameter has expired and FIFO is still full
- Full collision support:
  - Enforce collision (jamming and TX_ER assertion)
  - Truncated binary exponential backoff algorithm for random wait
  - Automatic frame retransmission (until retry limit is reached)
  - Retransmission from transmit FIFO following a collision
  - Automatic discard of incoming collided frames
  - Delay transmission of new frames for specified interframe gap
    - Programmable IFG duration for back-to-back frames
    - Supports Defer
- Performs framing functions
  - Preamble generation and stripping

  - Programmable preamble size on transmit side
  - Optional Recognition of user defined preamble, and placement of preamble as part of data
  - Optional Transmission of user defined preamble
- CRC generation and checking
- Framing error handling

- Automatic padding of short frames on transmit to 64 bytes
  - Global Mode
  - Frame Based
- Detection of all erroneous frames as defined by IEEE 802.3-2002
  - Filtering of erroneous frames
  - Statistics gathering
- Multi-buffer data structure
- Diagnostic mode:
  - Internal and external loopback mode
  - Echo mode (MII and RMII modes)
- Serial management interface MDC/MDIO
- 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
  - Excessive deferred frame indication
  - Error reporting modes
    - Per frame reporting in 10/100 (compatible with PowerQUICC II)
    - Interrupt based reporting for Gigabit rates
- Receiver network management and diagnostics
  - CRC error indication
  - Non-octet alignment error
  - Frame too short/long
  - Overrun
  - Busy (out of buffers)
- Frame Filtering and Address recognition
  - MPC82xx backward compatible filtering mode
    - Five 48-bit addresses recognized or 64-bin hash table for physical address
    - 64-bin Group Address Hash Table

  - Optional Broadcast address filtering
  - Promiscuous Mode. Receives all frames regardless of address.
- Programmable Parse Command Descriptor (PCD) mode
  - Programmable field Extraction: MAC src, MAC dst, VLAN Tag
  - Internal/External Lookup Tables with programmable size
  - Internal CAM Emulation or four way Hash based Lookup Modes
  - Programmable Match Actions: Receive, Reject

- Programmable maximum frame length
- Enhanced MIB statistics
  - Supports IEEE 802.3-2002 Layer management for DTE
    - IEEE 1GE Management Enhancements
    - MAC Entity managed Objects
    - MAC Control entity managed object
    - PAUSE entity managed objects
  - Supports IEEE 802.3-2002 Layer Management for Base Band repeaters
  - Supports IETF RFC 2819 / STD0059- RMON MIB
  - Supports IETF RFC 3635 Managed Object for Ethernet like Interface
  - Extensions to the standard statistics is added in order to adapt it to an environment with non-standard length (Jumbo frames, Tagged frames)
  - Enhanced transmitter statistics for non-CPU operation
- VLAN Support
  - Optional VLAN Tag extraction and insertion on Received Frames
  - Optional VLAN Tag insertion on Transmitted frames
  - Optional Enqueueing by VLAN priority field
  - Optional Filtering by VLAN Tag in Extended Parsing mode
- Supports IEEE Std 802.1p™/Q QoS with up to 8 Tx/Rx priority queues
  - Up to eight Tx/Rx BD rings to satisfy QoS requirements.
  - Receive queueing based on VLAN priority, IPv4 TOS field or IPv6 TC
  - Transmit Scheduler with SPQ/WFQ and Rate Limiter
- Optional L3 header checksum calculation
- Supports two Interrupt Modes
  - PQ2 compatible interrupts enabled on a per frame basis
  - Interrupt coalescing on Received frames.
  - Programmable threshold per receive queue
  - Programmable coalescing timeout
- Optional Shift of Data buffer by two bytes for L3 header alignments
- Extended Features

  - Queuing decision based on VLAN Priority or L3 IPv4 TOS field
  - IP header checksum verification and calculation
- Magic Packet Detection for Low Power Systems
- Support for Loss Less Flow Control

29.4 Functional Description

29.4.1 Ethernet Frame Transmission

29.4.1.1 Ethernet Tx Flow

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the UCC Ethernet controller (UEC) activates its transmit scheduler and begins to poll the first transmit buffer descriptor (TxBD) in one of the (up to) eight transmit queues as chosen by the scheduler. The TxBD ring is polled every 256 transmit clocks by default. The user can program the polling delay; refer to the UCC UTPT register for details). If TxBD[R] is set, the UCC Ethernet controller begins moving the contents of the transmit buffer from memory to the Tx Virtual FIFO. The Ethernet MAC transmitter takes data from Tx virtual FIFO and transmits the data through the appropriate interface (MII/RMII) to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected, unless a collision within the collision window occurs (in half-duplex mode) or an abort condition is encountered.

The UEC transmit scheduler schedules up to eight queues for transmission. The scheduler combines both strict priority queues (SPQ) and weighted fair queues (WFQ). The user can map latency-sensitive data transmissions to the strict priority queues and map bandwidth-dependent data transmissions to the weighted fair queues. In addition, the scheduler maintains a rate limiter for limiting the average bit rate transmitted by the Ethernet port.

If the user has a frame ready to transmit, setting the Transmit on Demand bit (UCC[TODR]) eliminates waiting for the next poll; the MAC immediately fetches the next buffer descriptor. The actual transmission on the line begins after all data for the frame is loaded into Tx virtual FIFO or sufficient transmit data (determined by the Tx FIFO threshold register) is in Tx Virtual FIFO. The MAC asserts TX_EN and sends the preamble sequence, start frame delimiter, and frame information in that order.

In half duplex mode, the transmitter waits for the carrier sense signal, CRS, to remain inactive for 60 bit times and transmission begins after an additional 36-bit times (96-bit times after CRS became active); refer to Table 29-1. When a station starts transmitting, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam of all ones on its frame and stops transmitting. Collisions usually occur close to the beginning of a frame. The station then waits a random period of time (backoff) before attempting to transmit again. Once the backoff completes, the station waits for silence on the LAN and then begins retransmission. This process is called a retry. If the frame is not successfully transmitted within 15 retries, an error is indicated.

Table 29-1. Ethernet Parameters Periods

	10 Mbps	100 Mbps	1000 Mbps
Transmit rate per byte	0. 8 us	0.08 uµs	0.008 uµs
Preamble plus SFD	6.4 us	0.64 µus	0.064 uµs
Interframe gap	9.6 µus	0.96 µus	0.096 uµs
Slot time	51.2 µus	5.12 uµs	4.096 uµs

In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap (96-bit times) regardless of CRS. If CRS continues to be asserted, MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in Tx FIFO is retransmitted 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 UEC implements automatic full-duplex flow control. If a flow control frame is received, the MAC does not send data until the pause duration is over. If the MAC is sending data when a pause frame is received, it finishes sending the current frame, then suspends subsequent frames (except a pause frame) until the pause duration is over. For the latter, the MAC starts the timer while transmission is still in progress. According to 802.3, Annex 31B, the pause period should start after the current transmit frame completes. This effectively shortens the desired PAUSE time, and it should not really be a problem from the user's perspective unless the pause period is very short. In such cases the pause time should be increased.

In addition, the UEC supports transmission of flow control frames (pause frames). Two mechanisms are provided: automatic flow control and CPU controlled flow control

The user can program the UEC to append FCS (32-bit CRC) automatically at the end of the frame. The following bits in the programming model (if set) enable CRC generation:

- TxBD[PAD/CRC] is set in the last Tx BD
- TxBD[TC] is set in the last Tx BD
- MACCFG2[PAD/CRC] is set
- MACCFG2[CRE] is set

Table 29-2 specifies the behavior of the UEC depending on the bit programming.

Table 29-2. Tx CRC and PAD Mode

TxBD[PAD/CRC]	TxBD[TC]	MACCFG2[PAD/CRC]	MACCFG2[CRE]	CRC	PAD (Frame Size Under 64 Bytes)
ignored	ignored	1	ignored	All frames	All relevant frames
0	ignored	0	1	All frames.	No
1	ignored	0	1	All frames	Padding of frames which have TxBD[PAD/CRC] set.

Table 29-2. Tx CRC and PAD Mode (continued)

TxBD[PAD/CRC]	TxBD[TC]	MACCFG2[PAD/CRC]	MACCFG2[CRE]	CRC	PAD (Frame Size Under 64 Bytes)
0	0	0	0	No	No padding
0	1	0	0	User-programmable per BD	No padding
1	ignored	0	0	Padding of frames with TxBD[PAD/CRC] set.	

Following the transmission of FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the frame if reached (TxBD[L]=1) the Ethernet controller updates the status bits in the Tx BD.

If TxBD[PAD/CRC] is set, the Ethernet controller pads any frame shorter than 64 bytes. Note that transmitter padding is done to 64 bytes regardless of the presence of a VLAN tag in the frame.

A graceful transmit stop is used to pause transmission. 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 UCCE[GRA] interrupt occurs once the graceful transmit stop operation is completed.

The UEC optionally inserts a VLAN Tag to the frame immediately after the MAC source address. The user may program the UEC to choose one of eight possible VLAN Tag values on a per frame basis. Note that some restrictions apply on the TxBD and data buffer when this mode is enabled. These restrictions are specified in the TxBD programming model.

The UEC supports Tx enhanced performance mode to improve packet processing rate performance. The performance enhancement mode is programmable and can be set in the UCC parameter RAM. When this mode is activated there is a restriction on the TxBD ring size: The size of the TxBD ring must be greater than the maximum number of BDs associated per frame

While the UEC is in 10/100 Mbps mode it sends the least significant nibble of each byte first.

29.4.2 Ethernet Frame Reception

The UCC Ethernet controller receiver is designed to work with little core intervention and can perform pattern matching, data extraction, address recognition, Ethernet type recognition, CRC checking, VLAN detection, short frame checking, and maximum frame-length checking.

After a hardware reset, the software driver initializes the Parameter RAM and the configuration register, issues an INIT Rx Tx Host Command, and then sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames.

The MAC hardware checks when RX_DV is asserted, it looks for the start of a frame by searching for a valid preamble/SFD (start of frame delimiter) header, which is stripped and the frame begins to be processed. If a valid header is not found, the frame is ignored. Part of the preamble is (optionally) user configurable.

If the receiver detects the first bytes of a frame, the UCC Ethernet controller begins to perform the frame recognition function (such as Filtering). Two modes are supported: MPC82xx backward compatible frame filtering, and Extended Frame Filtering mode. The Extended Parsing supports large memory based lookup tables, thus replacing the need for an external CAM device. As a result of the frame recognition function the UEC either receives or discards the frame. On received frames larger than 64bytes the user may enable header manipulation (insert/replace/remove of VLAN TCI field).

In MPC82xx mode the receiver can also filter frames based on physical (individual), group (multicast), and broadcast addresses.

If a frame is accepted, the UEC fetches Receive Buffer Descriptor (RxBD) from FIFO. If RxBD is not being used by the software (RxBD[E] is set), UEC starts transferring the incoming frame. RxBD[F] is set for the first RxBD used for any particular receive frame.

When the buffer is filled, the UEC clears RxBD[E] and, if enabled the UEC, 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 RxBD's are used; thus, no collision frames are presented to the user except late collisions, which indicates LAN problems.

The length of the data buffer associated with the RxBD, is determined by the MRBLR field in the UCC Parameter Ram. The smallest valid value is 128 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 and (optionally) the checksum fields are checked and written to the data buffer. If header manipulation is enabled, the value of the CRC written by the UEC at the end of the data buffer is not valid, and needs to be recalculated in case the frame is transmitted on an other port.

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.

After the receive frame is complete, the UCC Ethernet controller (UEC) sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller generates a maskable interrupt indicating that a buffer was received and is in memory. If Receive frame interrupt is unmasked, and the interrupt coalescing counter has expired, the UEC issues an interrupt at the end of the frame.

The UEC receives serial data least-significant nibble first.

29.4.3 Interframe Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-frame 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 frame. 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 frame is not successfully sent within a specified number of retries, an error is indicated.

The minimum inter-frame gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back frames with this minimum spacing. In addition, after waiting a required number of clocks (based on the backoff algorithm), the transmitter waits for carrier sense to be negated before retransmitting the frame. Retransmission begins 36 serial clocks after carrier sense is negated for at least 60 serial clocks.

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 for 10/100) If a collision occurs after 64-byte times, no retransmission is performed, UCCE[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 UPSMR[RSH] = 1.

NOTE

The inter-frame gap between back-to-back frames may be programmable (in the IPGIFG register)

The rate limiter in the scheduler increases the actual Inter Frame Gap, from the minimum allowed in the standard, thus limiting the average bit rate on the line.

29.4.4 Multithreading

The UEC processes frames at wirespeed rates. To enhance performance, the UEC receiver and the UEC transmitter are able to process a programmable number of frames at one time. This is implemented with the multithreading mechanism. Each thread processes a different frame. The required number of threads is 1 (normal) or 2 (optimized performance). The INITETHERNET command programs the data structures needed for multithreading. See Section 29.8, “Ethernet Command Set,” for more details.

29.4.5 Virtual FIFO

The UEC Receive and Transmit Virtual FIFOs are divided into blocks of 128 bytes each. With this FIFO configuration the UEC handles frames of up to 128 bytes in an optimal manner. The size of the Virtual FIFO is software configurable by programming the URFS and UTFS registers. See Section 26.5, “Fast Protocol FIFO Configuration Registers,” for suggested value of Virtual FIFO size. The size of Virtual FIFO allocated for the UEC Transmitter and for the UEC receive depends on the bit rate, frame size, and overall load of the system bus and the QUICC Engine block.

NOTE

For the Rx Virtual FIFO, the user MUST allocate a memory block of size URFS+8 bytes in the multiuser RAM. The minimal size for URFS and UTFS is 408 bytes. If 2 threads are used, the minimal size for UTFS is 816 bytes.

29.4.5.1 IP Header Checksum

The UEC may be programmed to check IP header checksum on the receiver, and generate IP header checksum on the transmitter.

On the receive side, if this feature is enabled in REMODER, the receive automatically checks the IP frame checksum, and reports an error in the RxBD. The UEC parses Ethernet frames (Etype > 0x600 programmable value) or 802.3 frames with a SNAP header. There are a number of restrictions on the types of frames the checksum is calculated on the receive side.

In termination mode (REMODER[IWEn]=0) (MPC82xx legacy mode or Extended parsing and filtering mode): IPchecksum check is operational only on uncomic frames without a VLAN Tag or on incoming frames which have one VLAN Tag.

On the transmit side, If enabled (in TEMODER) the user indicates to the UEC the offset from the beginning of the frame where the IP header starts. On a per frame basis (in the TxBD) it is possible to choose one out of eight user programmable offsets (programmed in the Tx Parameter RAM). Note that if the offset is programmed to 0xFF, checksum is not calculated. Also note that the value of the IP checksum in the IP header places by the CPU in memory must be zero. The IP header must reside in the first 128 bytes of the frame, and in the first data buffer (first TxBD).

The IPv4 header checksum is calculated by breaking up the header (including any options) into a sequence of 16-bit words, summing the words as a one s compliment sum, and performing a one s compliment (bit wise inversion) of the result. The header checksum field is initialized to zero in forming the original sum. One s complement summation is implemented most easily as a normal binary summation, but with the any carries out of the MSB being counted and summed back into the result until no more carries occur.

29.4.6 Flow Control

Flow Control is a mechanism for limiting the bit rate of the received frames when congestion is detected in the system. This is achieved by transmitting the Flow Control frame from the congested port. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 29-3 shows the flow-control frame structure.

Table 29-3. 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 period measured in quanta of 512 bits time most-significant octet first.

Table 29-3. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	Extension Field	—	As programmed in UEMPR register. See Section 29.5.1.17, "UCC Ethernet Mac Parameter Register (UEMPR)."
40	Reserved	—	—
4	FCS	—	Frame check sequence (CRC)

29.4.6.1 Transmitting Flow Control Frames

The UEC has two mechanism for transmitting flow control frames:

- Hardware initiated Flow Control frames
- Host CPU initiated Flow Control frames

The hardware initiated Flow Control frames occur when the when the UCC Receive FIFO has reached a programmable threshold (UCC receive virtual FIFO emergency Threshold Register—URFET, see Section 26.5.3, "Receive Virtual FIFO Emergency Threshold (URFET)." The MAC Parameter is user programmable. The hardware automatically transmits additional Flow Control frames when the MAC parameter has expired unless the UCC receive FIFO is below the threshold. Also, if the Virtual FIFO has reached its low threshold, the UEC automatically transmits a Flow Control Frame with its MAC parameter set to zero, thus enabling the other side to resume transmission. The Automatic flow control mode is be enabled by setting UPSMR[AUFC] bit to one. It is possible to send an other pause frame while the pause timer has not expired. Therefore when the Receive FIFO is below the threshold, the UEC sends a Pause Control Frame with MAC parameter set to zero.

For CPU controlled flow control, the CPU issues a START FLOW CONTROL command to the QUICC Engine block via the CECR command register. The CPU specifies the value of the MAC Parameter in the UEMPR[PT] register. In order to send a Flow Control frame with MAC Parameter set to zero, the CPU issues a STOP FLOW CONTROL command.

Note that if the UEC detects nested Control frames (such as CPU control frame nested in the automatic control frame mechanism), it does not automatically transmit the Flow Control frame with zero MAC parameter. In this event, when the receive FIFO is below the threshold, a zero MAC parameter Flow Control frame is automatically sent by the UEC, only after the CPU has initiated a zero parameter Control Frame. In other words; the UEC actually transmits a FC frame with MAC Parameter = 0 if two conditions are met:

- The CPU has issues a STOP FLOW CONTROL command
- The Rx FIFO is filled below the URFET threshold

29.4.6.1.1 Lossless Flow Control

The LossLess flow control feature is enabled by setting REMODER[LossLessFCEn] = 1. If enabled, the UEC transmits a Flow Control frame if the corresponding Rx BD Ring has reached a threshold. BDx LossLess FC Threshold contains the threshold for each one of the RxBDs in terms of the number of empty RxBDs. If the UEC detects a condition in which there are fewer empty RxBDs than programmed in the threshold, a flow control frame is automatically transmitted.

Every time the CPU handles an RxBD, it must update the pointer to the next RxBD in the entry corresponding to the RxBD ring in the LossLess Flow Control Table. The CPU must update the BDx CPU BD Pointer with the pointer to the Buffer descriptor of the first RxBD that was not handled. Note that if the RxBD has the 'W' bit set, the CPU then must place the pointer to the first RxBD in the RxBD ring. It is strongly advised to write the value of the pointer without any read transaction for the CPU (for minimal performance impact).

The CPU must set the 'BDx LossLess FC Threshold' to a value that ensures that there are always some empty RxBDs in the RxBD ring. The number of empty RxBDs are MTU/MRBLR + 5 + N. MTU is the Maximum Transfer Unit (max frame size). If MTU > MAXD1 or MAXD2 then the number of empty RxBDs is MTU/MAXD1 + 5 + N.

N is determined as follows: All the frames temporarily located in the Virtual FIFO must have an empty RxBD available. In the worst case condition the Virtual FIFO is filled with the minimum size frame (such as 64 byte frames). Therefore N = Virtual FIFO size / (64 + 8) bytes.

NOTE

This feature is not available in MPC832x silicon revision 1.

29.4.6.2 Receiving a Flow Control Frame

When flow-control mode is enabled (MACCFG1[Rx_Flow] is set) 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 pause-flow control frame can be sent (if MACCFG1[Tx_Flow] is set). Normal transmission resumes after the pause timer stops counting. If another pause-flow control frame is received during the pause, the period changes to the new value received.

All received pause-flow control frames are mapped to the default receive queue as programmed in TCI field in the Global Rx Parameter RAM.

29.4.6.3 Back Pressure

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 UEC 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. If the medium is idle, the UEC raises the 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, UEC 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). These transmitting-preamble-for-back-pressure collisions are not counted. If HAFDUP[BPNB] is set, the UEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If cleared, the UEC 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 UEC drops the carrier (ceases transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the UEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. HAFDUP[BPNB] applies for any collision that occurs during the sending of this packet. Collisions for packets while THBP is asserted are counted. UEC does not defer while attempting to send packets while in backpressure.

29.4.7 Frame Filtering and Address Recognition

The UEC supports two address recognition modes:

- MPC82xx backward compatible filtering mode
- Extended parsing mode

29.4.7.1 MPC82xx Compatible Address Filtering Mode

In the MPC82xx backward-compatible filtering mode (REMODER[EXP]=0) the UCC Ethernet controller (UEC) implements MPC82xx compatible filtering mode based on Destination MAC Address. In this mode the UEC filters 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 29-6 is a flowchart for address recognition on received frames. In the physical type of address recognition, the UCC Ethernet controller (UEC) first compares the MAC destination address field of the received frame with the physical address that the user programed in MACSTNADDR1,2 registers. If the comparison fails, and if REMODER[EXF]=1 (extended features are enabled) the UEC compares the MAC destination address field to the PADDR_H1..4]and PADDR_L1..4 entries programmed by the user in the Rx global parameter RAM. It is possible to change the values of PADDR1..4 on the fly. The UEC Rx filtering behavior is unpredictable during the time that these values are changed.

If it fails, the controller performs address recognition on multiple individual addresses using the IADDR_H/L hash table.

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.

See Address filtering flow in Figure 29-6.

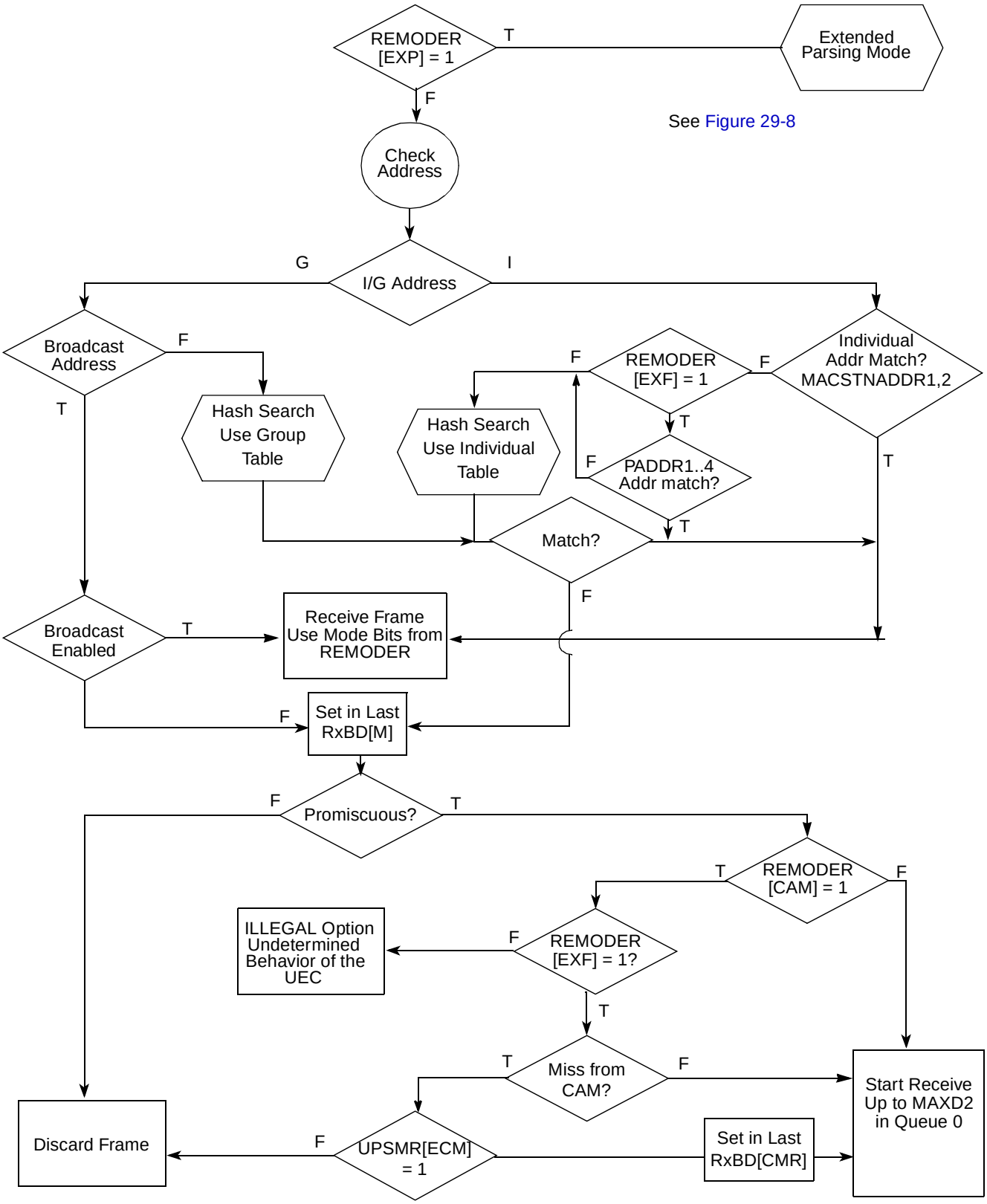


Figure 29-6. Address Filtering Flow REMODER[EXF]=0

29.4.7.1.1 Valid Bit 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.

In order to program the GADDR field the user executes a QUICC Engine command named SET GROUP ADDRESS. As a result the UCC 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 0 and 63. Bit 26 of the CRC result selects between the two GADDRs or IADDRs; 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 8 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 8 group and 8 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.

29.4.7.2 Extended Parsing Mode

29.4.7.2.1 Introduction

The Extended Parsing mode allows for enhanced Frame filtering based on fields extracted from the L2 of the frame. The choice of the fields is user programmable. This mode is selected by the user by programming REMODER[EXP]=1.

The CPU initializes eight byte data structures called Parse Command Descriptors (PCDs). In the PCDs the CPU selects which header fields are to be extracted from the frame headers to generate a LookupKey, and the type of LookupTable to be used for the lookup. The PCDs are programmed by the CPU at initialization.

Upon arrival of a frame, the UEC uses the PCDs as directives to parse the frame headers, generate a LookupKey (one or more) and perform table lookups. The LookupKey is generated by concatenating header fields. Note that frames of size under 64 bytes that are received by the UEC (as programmed in UPSMR[RSH] and MINFLR), are enqueued in queue 0 without parsing al all.

29.4.7.2.2 High Level Description of Parse Command Descriptors (PCDs)

There are different types of PCDs which are distinguished by their opcode field. The following sections describe the types of PCDs available. The first PCD are used to generate or modify the LookupKey:

- The GenerateL2LookupKey PCD is used to parse the frame and extract L2 header fields. The user may select the header fields to be extracted by setting an enable bit for each header field. The selected header fields are concatenated into a LookupKey which is used to perform a table lookup.

The following header fields may be selected for extraction: MAC destination address, MAC source address, VLAN TAG. Up to 16 bytes from the frame headers may be extracted.

- The change mask PCD is used to mask fields from a LookupKey. For example, part of the TCI fields may be masked. The original LookupKey is saved in temporary storage.
- The restore mask PCD is used to restore the original LookupKey, to allow a new lookup based on other fields.

The Lookup PCD description follows:

- Four/Eight Way Hash Lookup PCD is used for long LookupKeys (up to 1624 bytes and/or for large lookup tables. In this mode the LookupKey is hashed and the result is used to index a LookupTable in internal or external memory. The table is organized in four way sets. If all four ways are filled the user may add an other Lookup Table containing an other four ways set (applicable to Eight Way Lookup PCD). A mismatch in the compare of all four (or eight) tags, yields to a table lookup miss. Figure 29-7 is a description of the lookup flow using this mode:

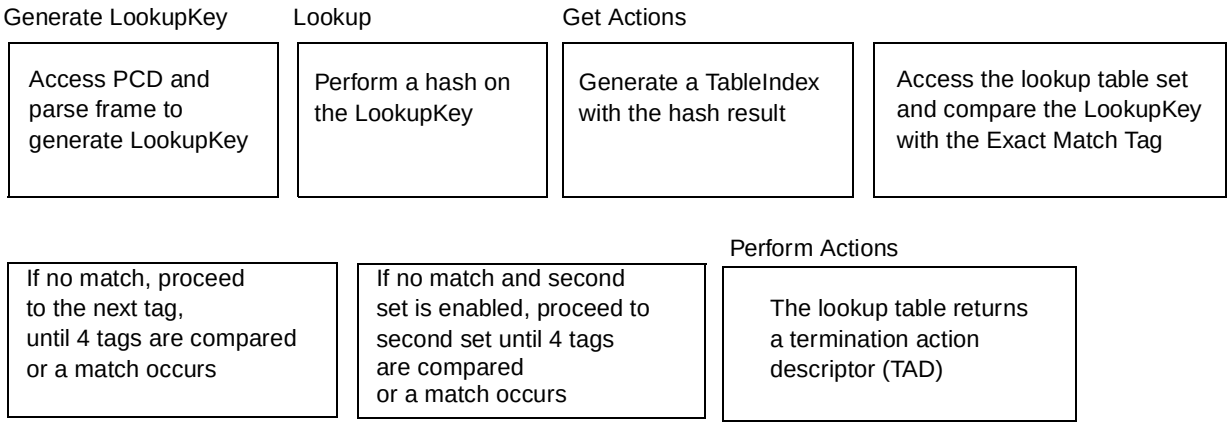


Figure 29-7. Flow for Hash mode

29.4.7.2.3 Address Filtering Flow

The following figure depicts the flow executed by the UEC in Extended Parsing mode:

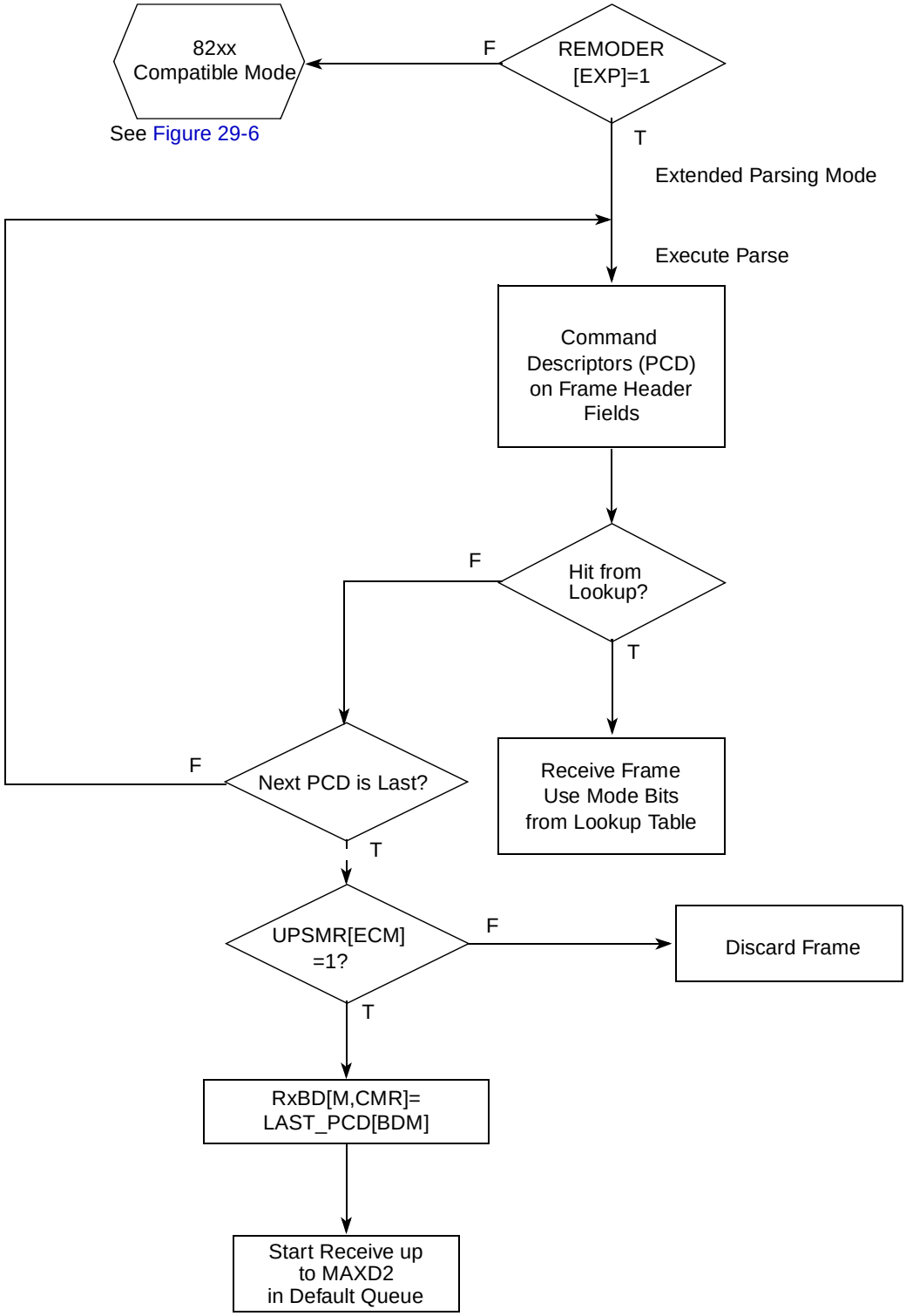


Figure 29-8. Address Filtering Flow REMODER[EXF]=1

29.4.8 Header Manipulation

The UCC Ethernet controller (UEC) supports user programmable insertion, replacement and deletion of a 2 byte Ethernet Type of 0x8100 and two bytes of VLAN Tag and priority on the Receive direction. On the transmit direction only the insertion operation is supported.

In the receive direction, if in Header Modify mode enabled by programming REMODER[VTagOP,VNonTagOP] bit not equal to zero (or if in Extended Parsing mode the corresponding bits in the result from the lookup table are non equal to zero) and if the frame does not contain a VLAN tag already, the UEC inserts a user defined VLAN Tag to the frame immediately after the MAC source address. The value of the VLAN Tag is taken from a user programmable default (if REMODER[EXP]=0) or it is programmed by the user in the lookup table, to allow different VLAN tags depending on the type of frame. This scheme allows for allocation of a different VLAN Tag for different Ethernet type fields.

If Header modify mode is enabled the last four bytes in the received frame do NOT contain the valid CRC of the modified frame.

In the Transmit direction, the user enables VLAN Tag insertion by programming the VLAN ID in the Buffer Descriptor. One of eight possible programmable VLANs is chosen. VLAN Tag ID=0, means no VLAN Tag insertion.

Note that header manipulation is not performed on received frames that are shorter than 16 bytes.

29.4.9 Receive and Transmit Buffer Descriptors (BDs)

The ethernet controller maintains multiple Tx/Rx Buffer Descriptor (BD) rings to satisfy QoS requirements.Up to eight Receive and eight Transmit BD rings are supported.

29.4.10 Quality of Service (QoS)

29.4.10.1 Receiver Queueing Decision

The queueing decision is based on the value of the QoS fields in the frame headers as they are received from the line (that is, before any header manipulation performed by the UEC), or the result of the lookup. The QoS fields that affect the queueing decision is taken from a global user programmable setting, or from the lookup table, or from the VLAN priority field in the VLAN header, or the TOS field in the IPv4 header or the TC field in IPv6 frame. Table 29-4 specifies how the queue priority is chosen by the UCC Ethernet controller.

Table 29-4. Priority Selection Mode

User Programmble Mode if REMODER[EXP]=0 REMODER[RQoS] is used if REMODER[EXP]=1 TAD[RQoS] is used	Incoming Frame type			
	VLAN Tagged not IP	**VLAN Tagged and IPv4 or IPv6**	**not VLAN Tagged not IPv4 or IPv6**	**not VLAN Tagged and IP**
Use default queue REMODER[RQoS]=00 or TAD[RQoS]=00	User Programmable Queue as programmed in VPriority field in TCI or in TAD.			
Use L2 Priority REMODER[RQoS]=01or TAD[RQoS]=01	VLAN priority	VLAN priority	Default Queue	Default Queue
Use L3 Priority REMODER[RQoS]=10 or TAD[RQoS]=10	VLAN priority	IP TOS or Traffic class	Default Queue	IP TOS or Traffic class

Note:

1. In MPC82xx Filtering mode (REMODER[EXP]=0): RQoS field is taken from the REMODER register. In Extended Parsing mode (REMODER[EXP]=1): RQoS field is taken from the Lookup Table (LUT) Entry.
2. Default queue is user programmable Rx Global Parameter RAM TCI[VPri].
3. Frames shorter than 64 bytes are enqueued in queue 0.

In Extended Parsing mode (REMODER[EXP]=1) there is also the option of explicitly determining the queue number in the lookup table, overriding the value of the VPri to TOS fields in the frame headers. The following figure depicts the queuing decision in term of a flow:

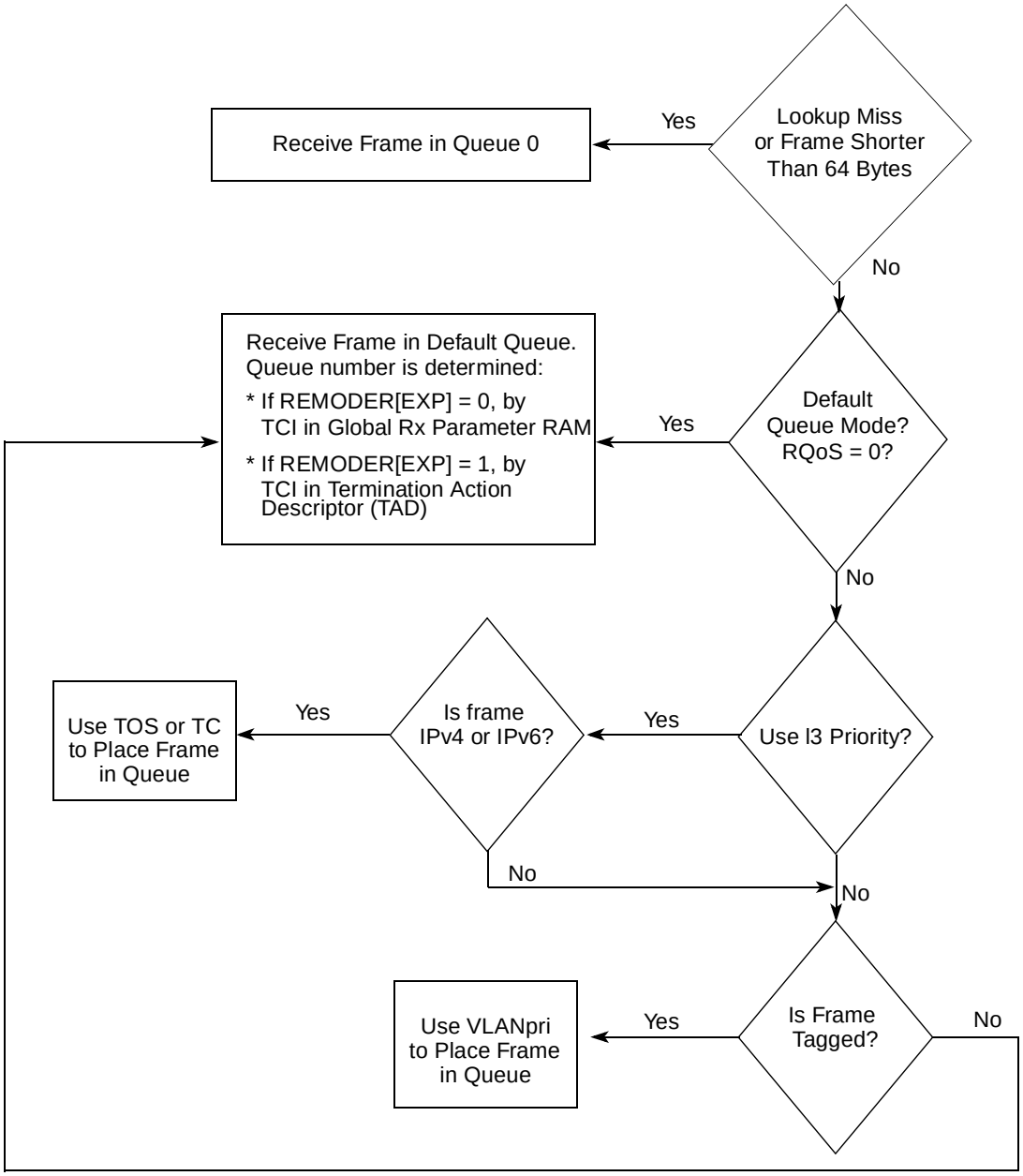


Figure 29-9. Receive Queueing Decision

29.4.11 Receive Interrupt Coalescing

Interrupt coalescing feature applies to interrupts issued on received frames. The user can program the UEC to issue a receive frame interrupt every programmable number of frames (programmed in Interrupt Timeout entry in the Global Rx Parameter RAM).

The ethernet controller may also generate interrupts after every Buffer Descriptor is received or transmitted, if enabled in the Buffer Descriptor and unmasked in the Interrupt Event Register. There is no interrupt coalescing for Buffer Descriptor interrupts.

29.4.12 Align IP address

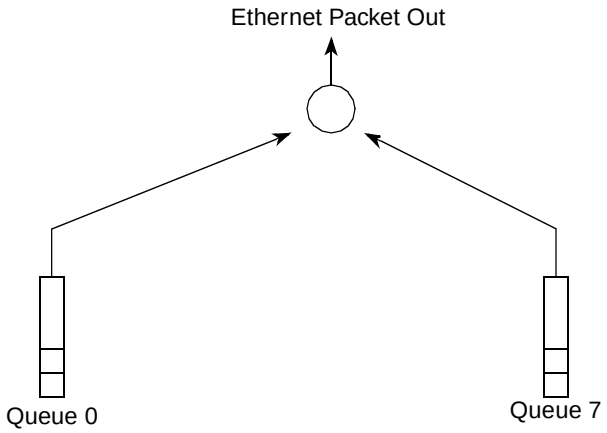
The UCC Ethernet controller (UEC) supports IP address alignment to four bytes on received frames. If enabled, the UEC places the first byte of the frame two bytes from the address pointed by the pointer in the RxBD. The first two bytes in the data buffer contain undefined data.

29.4.13 Statistics Gathering

The UCC Ethernet controller (UEC) implements a set of statistical counters. These counters allow the implementation of MIBs to support IEEE 802.3-2002,1Giga Ethernet Enhancements RFC2819 and RFC3635. Some of the counters are implemented in Hardware, and they have a maskable event bit that interrupts the CPU when they overflow. Other counters have their value stored in the Parameter RAM. The receive statistics counters and the transmit statistics counters my be disabled separately in order to improve overall performance of the UEC.

29.4.14 Ethernet Scheduler Theory of Operation

The scheduling scheme of the Ethernet controller transmitter role is to select the next packet to be transmitted on the line. A single transmitter resource (UCC Ethernet port) is shared by 8 queues, as depicted in the figure below. Each of the queues contains a random number of packets with a random length, from 64 bytes to Jumbo frame (up to 9600 Bytes). The scheduling system is invoked whenever a new packet has to be transmitted, and updated after packet transmission completion.



Ethernet scheduler role is to determine the next packet to be transmitted on the shared transmitter.

Figure 29-10. Ethernet Scheduler Role

29.4.14.1 Scheduling System Features

- Up to 8 Output Queues
- Strict priority (SP) scheduling
- Weighted Fair queuing (WFQ) scheduling
- Combined mode of both SP and WFQ in the same queue Structure

- Token bucket type shaping
- Rate limiting
- Various Traffic Shaper bypass modes, global and per queue for high QoS
- Transmit ASAP mode
- Extra bandwidth mode
- Supports both Work conserving and Non work conserving operation
- Maximum packet length of Jumbo size (9.6KB)
- BW Resolution in WFQ of 4Mbps
- Minimum rate of 4Mbps (Excluding rate of 0—Queue disabled)
- Maximum rate of 100Mbps continuously

Minimum rate limiting 32kbps with TSRUnit=1us, and can be even lower with higher value of RTSR pre scalar please see formulas below.

Note that is the scheduler uses timer stamp register number 2. See Section 19.3.10, “QUICC Engine Time-Stamp Registers (CETSRn).”

29.4.14.1.1 Scheduler Description

The Ethernet scheduling system is composed of two main functional modules: A work conserving scheduler, and a data shaper. This is shown in the next figure:

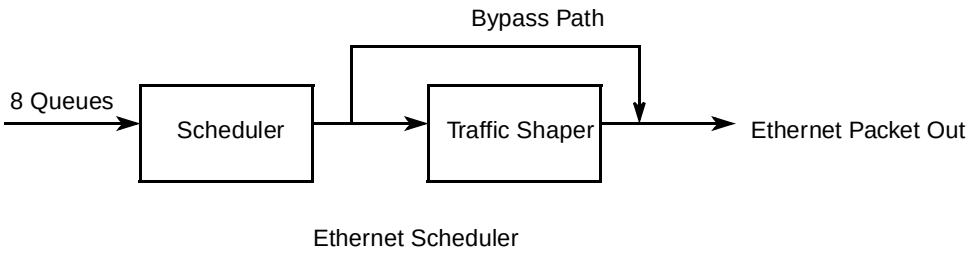
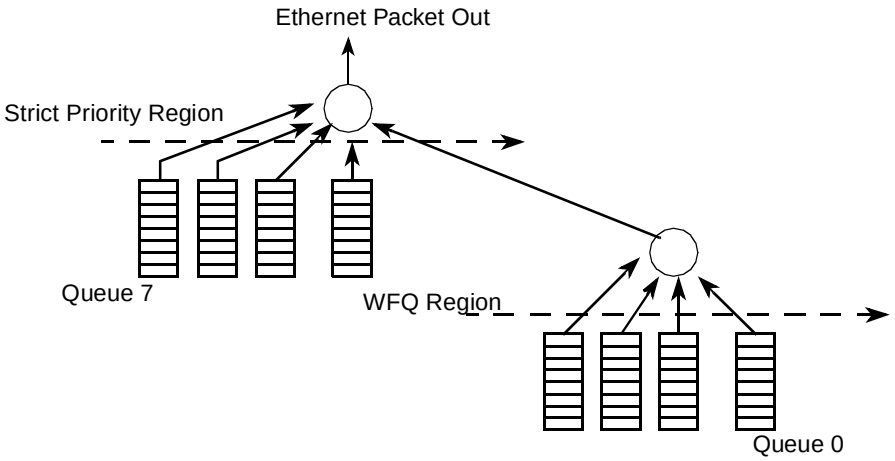


Figure 29-11. Ethernet Scheduling System

The scheduler module selects the next queue to be handled by an Ethernet UCC. The Traffic Shaper controls the data rate on the line using a real time scale, enabling the user either to limit data rate to a certain value, or to limit the maximum burst length, or both.

29.4.14.1.2 Scheduler Module

The scheduler module supports two scheduling hierarchies. The upper hierarchy imposes a scheduling policy of Strict priority. The lower hierarchy implies a scheduling policy of Weighted Fair Queuing (WFQ). The scheduler module handles up to 8 queues, each of them may be associated either with a different QoS or pre allocated BW. A sample configuration is shown in the figure below:



Two-Hierarchy Configuration of the Scheduler Module

Figure 29-12. Sample Scheduler Configuration

In the example, queues 7–4 are transmitted first, if they are back logged. 3–0 that are configured in WFQ mode, are selected for transmission only if the strict priority queues are not back-logged. In this case, they are sharing the remaining bandwidth according to pre-defined relations.

Any queue may be configured either as SP queue (Strict priority) or as a WFQ queue. The SP queues are always with higher priority than the WFQ queues. Queues configuration can be varied on-line without any packet loss.

29.4.14.1.3 Traffic Shaper Module

The Traffic Shaper controls the absolute data rate on the bus. It can work in a mode of rate-limiting, Token bucket shaping, Bypass, or per queue by pass.

29.4.14.1.4 Toke Bucket Shaping

In this mode, the traffic shaper shapes output traffic according to the constraints of average data rate, and max burst length. Note that in Ethernet one must burst always in peak data rate until the end of the packet. Thus, the minimal value of maximum burst length is the longest packet, that may be a Jumbo packet in certain cases.

However, if the receiver side can tolerate longer bursts, the Traffic Shaper may allow a concatenation of packets (with minimum IPG as required by IEEE 802.3-2002 between them) in order to increase line efficiency.

Thus, the Traffic Shaper supports Token Bucket shaping with peak rate of 100Mbps, and programmable max burst length and programmable average rate.

29.4.14.1.5 Rate Limiting

In this mode, the traffic shaper adjust the IPG after any packet transmission so that the average data rate will be as programmed. For instance, if the required rate is 0.5Gbps, after a packet of length of 64 bytes

there will be an IPG with size of 64 bytes. This allows minimal FIFO size in the communication peer receive side.

29.4.14.1.6 Traffic Shaper Bypass

The traffic shaper can be canceled. In this mode, the scheduling is work-conserving, based on the scheduler module only.

29.4.14.1.7 Traffic Shaper per Queue Bypass

For any of the queues, two orthogonal Bypass modes are supported: Transmit ASAP and Extra BW. Any queue may be configured in each of them in orthogonal manner.

In “Transmit ASAP” bypass mode, the queue transmission condition is not depending on the Traffic Shaper state, such as transmission occurs instantly. This mode is suitable for highest priority, lowest absolute delay queue.

In “Extra BW” mode (Extra bandwidth mode), the BW a queue has is added to the nominal BW. For instance, assume that queues 5–0 are configured in WFQ, queues 7–6 in SP. Queue 7 is for high priority control packets, with small delay and small BW. Queue 7 is configured in Transmit ASAP bypass mode, and Extra BW bypass mode. Queue 6 is configured in Transmit ASAP bypass mode. Queues 5–0 are with no bypass mode.

The system behaves as follows: If the Traffic Shaper is programmed to 50Mbps, and the traffic in queue 7 is 10 Mbps, the total BW of the line is 50Mbps+ queue 7 rate that is 60Mbps. However, the traffic in queue 6 + queues 5–0 are limited to 50Mbps by the Traffic Shaper.

Queue 6 transmits instantly because it is in the Transmit ASAP mode; however, its BW is deduced from the total BW of 50Mbps.

Note that if the total BW of the “Transmit ASAP” queues is higher than the total BW budget, then the “Transmit ASAP” has priority (BW is not maintained).

See Section 29.15, “Traffic Shaper Programming Considerations,” for example of scheduler programming.

29.4.14.1.8 Dynamic Changes

The queue configuration may changed by the user on-line without ceasing data transmission. In order to transfer a queue from the SP group to the WFQ group the user can change the value of the bit associated with this specific queue in the StrictPriorityQ parameter.

If a queue is transferred from the SP group to the WFQ group, the parameter WeightFactor should be initialized as well.

If the data rate of a certain queue in the WFQ group has top be changed, the user may access directly the WeightFactor associated with this queue and change it. There is no need to stop transmission, neither stop substitution of packets in any of the queues.

29.4.14.1.9 Prioritized Queues

A Queue may be prioritized in two ways: Either by configure it to Strict Priority, or by giving it a very low WeightFactor in the WFQ system. In the first case, the queue will always gets its priority if it is not empty. The second case, however, will prevent the case of starvation if there is a chance that the SP Q will be busy most of the time. It is recommended that if a Queue that has to be in high priority is empty most of the time, it will be configured as SP queue.

If a Queue that has to get high priority is not empty most of the time, it is recommended to configure it as WFQ queue with low WeightFactor in order to prevent starvation. In this case, the TxASAP and ExBW bits should be configured as required by the application. TxASAP set means that whenever this queue is selected by the WFQ mechanism it will be transmitted, regardless of the state of the rate limiter. However, configuring it as WFQ will allow also other queues to be active.

See Section 29.5.3.3.3, "Scheduler Programming Model and Data Structures," for the scheduler programming model.

29.4.15 Magic Packet Detection

The UEC supports automatic detection of Magic packets. This feature is useful for Low Power systems. The Magic Packet feature is enabled by performing the SW sequence described below. Once the UEC enters magic packet mode no frames are received by the UEC until a magic packet frame is detected. When such a frame arrives the UEC automatically sets UCCE[MPD] and disables MACCFG2[MPE] bit. If unmasked an interrupt is issued to the CPU which wakes it up from low power mode.

The CPU must follow this sequence in order to enter Magic Packet Detection Mode:

1. The host performs a Graceful Receive Stop and Graceful Transmit Stop on the ethernet controllers.
2. The host determines that Transmission/Reception has stopped
3. The host disable UCC Tx and UCC Rx (disable ENT & ENR in the GUMR mode register).
4. The host sets a Magic Packet mode bit in the MAC (MACCFG2)
5. The host enable UCC Tx and UCC Rx (enable ENT & ENR in the GUMR mode register).

Now the UEC is in Magic Packet detect mode. Upon detection of a magic packet the UEC asserts an interrupt to the CPU.

1. The host determines that magic packet is detected by the Rx MAC (via interrupt)
2. The host performs a Resume Receive and Resume Transmit commands on the Ethernet controllers

29.4.16 UEC Physical Interfaces

The physical interface is user programmable by configuring the PSMR, MACCFG2 registers.

29.4.16.1 10 and 100 Mbps MII Interface Operations

The MII is the media independent interface defined by IEEE 802.3 for 10/100 Mbps operation. The speed of operation is determined by the TX_CLK and RX_CLK pins which are driven by the transceiver. The

actual bit rate of the transceiver is determined either by auto-negotiation or it is controlled by software via the serial management interface (MDC/MDIO pins) to the transceiver.

29.4.16.2 10 and 100 Mbps RMII Interface Operations

The Reduced Media Independent Interface (RMII) is defined by the RMII Consortium for 10/100 Mbps operations. In this mode, the UCC Ethernet controller converts MII signals to/from RMII signals. In RMII mode, a single clock reference (ref_clk) is sourced from the MAC to PHY (or from an external source)

29.4.17 Diagnostic Modes

29.4.17.1 Internal Loopback Mode

The UCC Ethernet controller operates in internal loopback mode by programming the GUMR[DIAG] bits. In this mode the Transmit output of the MII is connected internally to the receiver input, and RXD is ignored.

In RMII mode, loopback operation may be achieved in the same way as for MII, or by setting UPSMR[RLPB] bit. This bit allows for the loop to be closed in the RMII interface itself.

29.4.17.2 Echo Mode

In this mode, the channels automatically retransmit received data. The receiver operates normally. The UCC Ethernet controller will operate in echo mode by setting GUMR[DIAG] = 10.

For echo and loopback at the same time set GUMR[DIAG]=01 and UPSMR[RLPB]=0.

Note that Echo mode is supported only in MII and RMII mode

29.4.17.3 Full- and Half-Duplex Operations

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. When configured in half-duplex mode (half-duplex mode is only for MII/RMII 10/100 Mbps operations), (MACCFG2[FDX] register, bit 31), the MAC complies with the IEEE CSMA/CD access method. When configured in full-duplex mode (10/100 Mbps operation, (MACCFG2[FDX] = 1), the MAC supports flow control. When flow control is enabled, it allows the MAC to receive or send PAUSE frames.

29.5 Programming Model

The UCC Ethernet controller device is programmed by a combination of mode registers and of parameter RAMs. All accesses to and from 32 bit registers must be via 32-bit accesses. There is no support for accesses other than 32-bit on these registers. The following sections describe the registers in details.

29.5.1 Register Descriptions

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 IEEE 802.3. All of the 32 bit MAC registers must be accessed as 32 bit entities.

29.5.1.1 UCC Protocol Specific Ethernet Mode Register (UPSMR)

In Ethernet mode, the UPSMR, shown in Figure 29-13, functions as the Ethernet mode register.

NOTE

This register must be initialized after the GUEMR; see Section 22.3.2, "General UCC Extended Mode Register (GUEMR)."

offset	0x4															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—					ECM	HSE	0	Res	PRO	RES	RSH	RPM	R10M	RLPB	TBIM
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	AUFC		1	RMM	MDCP	RES	BRO	—								
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 29-13. UCC Ethernet Mode Register

Table 29-5. UPSMR Ethernet Field Descriptions

Bits	Name	Description
0–4	Reserved	Set to zero
5	ECM	This bit is valid if REMODER[EXP]=1 0 - Discard frames that miss in the Extended Parsing Mode Lookup 1 - Receive Frames that miss in the Extended Parsing Mode. The Rx queue is queue number 0.
6	HSE	Hardware Statistics Enable 0 - Do no perform hardware statistics. See Section 29.7.5, "UCC Statistics (Hardware Counters)." 1 - Perform hardware statistics. This bit is equivalent to the RMON bit in MPC82xx.
7	Res	Must be set to zero.
8	Res	Set to zero
9	PRO	Promiscuous valid if REMODER[EXP]=0. 0 Check the destination address of incoming frames 1 Receive the frame regardless of its address.
10	RES	Set to Zero.

Table 29-5. UPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
11	RSH	Receive short frames 0 Discard short frames (frames smaller than the value specified in MINFLR) 1 Receive short frames
12	RPM	Note: This field Must be cleared for 832x.
13	R10M	RMII 10/100 mode. See Table 29-6 for usage of this bit.
14	RLPB	RMII Loopback mode 0 - RMII is not in loopback mode 1 - RMII is in internal loopback mode Note that GUMR[DIAG]=00 in this mode.
15	TBIM	Note: This field Must be cleared for 832x.
16–17	AUFC	Automatic Flow Control Enable 00 - No Automatic Flow Control frames are sent by the UEC. CPU may initiate a transmit floe control frame issuing START FLOW CONTROL command. 01 - The UEC sends a pause frame when RxFIFO reaches its emergency threshold. See Section 26.5.4, “Receive Virtual FIFO Special Emergency Threshold (URFSET).” In half-duplex mode the MAC applies back pressure algorithm on the line by sending preambles on the line when no data is available for transmit. 10 - Reserved (not allowed) 11 - The UEC send a pause frame when the Receive buffers are busy. The UEC fills its RxFIFO until it reaches its threshold, and then a flow control frame is sent automatically. This mode, serves as an automatic flow control mechanism for external buffers. In half-duplex mode the MAC applies back pressure algorithm on the line by sending preambles on the line when no data is available for transmit. Note: When Rx_FIFO is out of the emergency threshold the MAC automatically sends pause frame with MAC parameter duration of “zero pause quantas” in full-duplex mode or disables the back pressure algorithm in half-duplex mode.
18	Res (1)	Must be set to ONE.
19	RMM	RMII Mode. See Table 29-6 for usage of this bit.
20	MDCP	Management Data Clock Prescale This field allows to MDC source clock to be divided by 8 0 - The source clock for MDC division is (CE/16 clock) 1 - The source clock for MDC division is (CE/2 clock) Note: If MDCP = 1, MIIMCFG[29:31] values should not be set to 000, 001, or 010. This bit is cleared by default. MDC divider - see MIIMCFG[28:31] Table 29-13
21	RES	Set to zero
22	BRO	Broadcast address 0 Receive all frames containing the broadcast address 1 Reject all frames containing the broadcast address unless UPSMR[PRO] = 1
23–31	Reserved	Set to zero

Table 29-6 summarizes the register programming needed to configure the UEC in the desired mode. Only the combinations in this table are allowed:

Table 29-6. Interface Mode Configuration

Mode	MACCFG2[IFM]	UPSMR[RPM]	UPSMR[TBIM]	UPSMR[R10M]	UPSMR[RMM]	MACCFG2[FDX]
MII	01	0	0	0	0	1 or 0
RMII(100)	01	0	0	0	1	1 or 0
RMII(10)	01	0	0	1	1	1 or 0

The behavior of the UCC Ethernet controller is not specified if programmed to other combinations of these bits.

29.5.1.2 Ethernet Event Register (UCCE)/Mask Register (UCCM)

The UCCE is used as the Ethernet event register when the UCC 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 UCCE bit. Interrupts generated by this register can be masked in the UCCM, which has the same bit format as UCCE (see Figure 29-14). Setting a UCCM bit enables and clears a bit masking the corresponding interrupt in the UCCE.

The UCCE can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values. Unmasked UCCE bits must be cleared before CP clears the internal interrupt request.

Offset	UCCx_base + 0x10 UCCE UCCx_base + 0x14 UCCM															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MPD	SCAR	GRA	CBPR	BSY	RXC	TXC	TXE	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	RXF7	RXF6	RXF5	RXF4	RXF3	RXF2	RXF1	RXF0
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 29-14. Ethernet Event/Mask Register (UCCE/UCCM)

Table 29-7 describes the UCCE/UCCM fields.

Table 29-7. UCCE/UCCM Mode Register Descriptions

Bits	Name	Description
0	MPD	Magic Packet Detection 0 - No Magic packet has been detected 1 - Magic Packet has been detected.
1	SCAR	Hardware Statistics Carry overflow Event. 0 - No overflow 1- A bit in the SCAR is set, and the respective bit in the SCAM register is set (unmasked).
2	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.
3	CBPR	In half duplex mode Change Back Pressure 0 - No change in backpressure state 1 - The UEC Transmitter is in backpressure state. The UEC is forcing preambles on the line, or is transmitting valid frames on the line. 1 - The UEC transmitter has entered the backpressure state (such as the receiver has signalled to the transmitter to enter backpressure. See Section 26.5.4, “Receive Virtual FIFO Special Emergency Threshold (URFSET)” for FIFO thresholds that triggers this event); OR the UEC has exited the backpressure state (such as the RxFIFO is filled below the threshold programmed in the URFET register, see Section 26.5.3, “Receive Virtual FIFO Emergency Threshold (URFET).”The status on the UEC transmitter is reflected in the UCCS Register. In full duplex mode Change Pause0 - No change in PAUSE state. 1 - The UEC transmitter has entered the PAUSE state (such as it has received a PAUSE frame and its transmitter is currently not transmitting on the line); OR the UEC has exited a PAUSE state (such as either the MAC parameter time has expired, or a PAUSE frame with MAC parameter zero has been received). The status on the UEC transmitter is reflected in the UCCS Register.
4	BSY	Busy condition. Set when a frame is received and discarded due to a lack of free Rx Buffer Descriptors
5	RXC	RX Flow 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.
6	TXC	TX Flow control. An out-of-sequence frame was sent.
7	TXE	Tx error. An error occurred on the transmitter channel. In full duplex mode this is an underrun condition.
8–15	TXB0–7	Tx buffer. Set when a buffer has been sent on the Ethernet channel, from queue number 0–7 respectively. Note that interrupts for TxBDs are asserted to the CPU after the Last TxBD of the frame has been transmitted.
16–23	RXB0–7	Rx buffer. Set when a complete buffer is received on the Ethernet channel on queues number 0–7 respectively and the respective RxBD[I] bit is set.
24–31	RXF0–7	Rx frame. Set when a complete frame is received on the Ethernet channel on queue number 0–7 respectively.

29.5.1.3 Ethernet Status Register (UCCS)

The UCCS contains the real time status of Ethernet Transmitter.

Offset	UCCx_base + 0x17 UCCS							
Bits	0	1	2	3	4	5	6	7
Field	0						BPR	MPD
R/W	R							

Figure 29-15. Ethernet Status Register (UCCS)

Table 29-8 describes the UCCS fields.

Table 29-8. UCCS Register Descriptions

Bits	Name	Description
0–5	—	Read zero
6	BPR	Back Pressure (in half duplex mode) 0 - UEC is not in backpressure state 1 - The UEC Transmitter is in backpressure state OR Pause State (in full duplex mode) 0 - The UEC is not in PAUSE state. 1 - The UEC transmitter is in PAUSE state and is not transmitting on the line.
7	MPD	Magic Packet Detected 0 - No Magic Packet was detected 1 - Magic Packet was detected

29.5.1.4 MAC Configuration 1 Register (MACCFG1)

MACCFG1 and 2 provide the means to configure the MAC in multiple ways:

- Adjust the preamble length from the nominal 7 bytes to some other (non-zero) value.
- Vary pad/CRC combinations using three different pad/CRC combinations to handle a variety of system requirements. For frames that already have a valid FCS field, 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-frame basis.

The MACCFG1 register is written by the user. Figure 29-16 describes the definition for the MACCFG1 register.

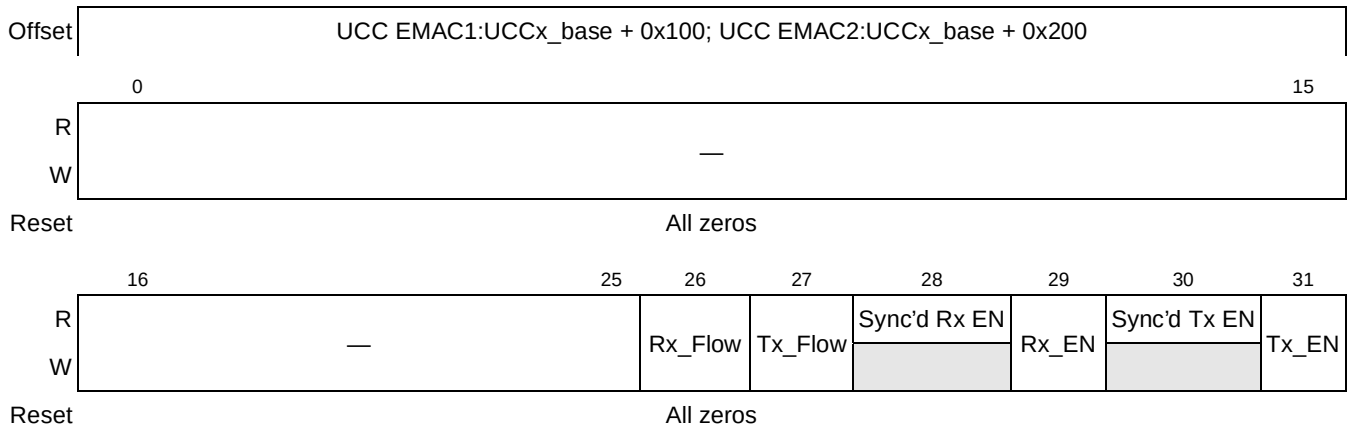


Figure 29-16. MAC Configuration 1 Register (MACCFG1)

Table 29-9 describes the MACCFG1 fields.

Table 29-9. MACCFG1 Field Descriptions

Bits	Name	Description
0–25	—	Set to zero
26	Rx_Flow	Receive flow control. 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 control. 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. 0 The MAC may not receive frames from the PHY 1 The MAC may receive frames from the PHY. This bit must be set before GUMR[ENR] is set.
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. 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system. This bit must be set before GUMR[ENT] is set.

29.5.1.5 MAC Configuration 2 Register (MACCFG2)

MACCFG2 is written by the user. Figure 29-17 describes the definition for the MACCFG2 register.

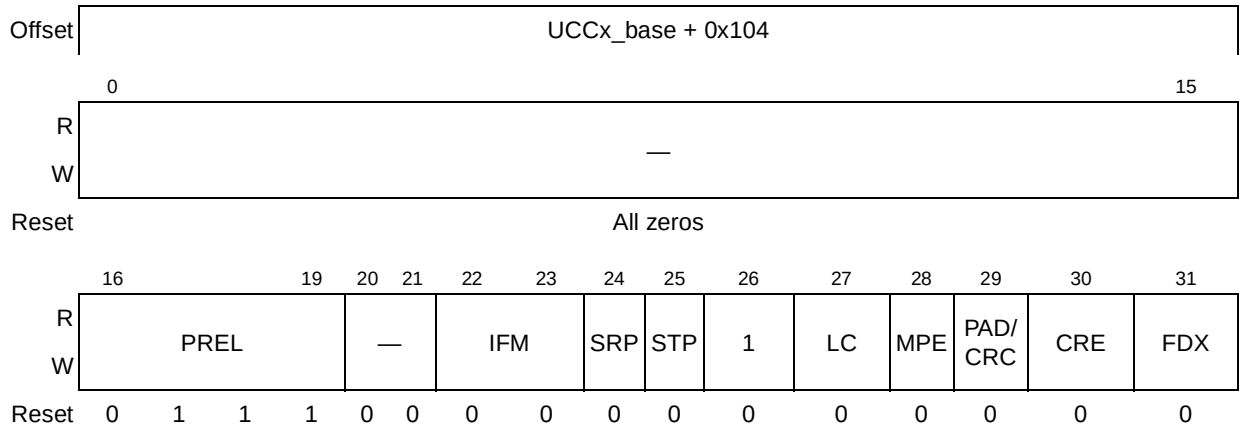


Figure 29-17. MAC Configuration 2 Register (MACCFG2)

Table 29-10 describes the MACCFG2 fields.

Table 29-10. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	Reserved	Set to zero
16–19	PREL	Preamble Length This field determines the length in bytes of the preamble field of the frame. Its default is 0x7. A preamble length of 0 is not supported. The Tx MAC supports only MACCFG2[PREL] = 3..7.
20–21	Reserved	Set to zero
22–23	IFM	Interface Mode. This field determines the type of interface to which the MAC is connected. Its default is 00. See Table 29-6 for usage of this bit. 00 Reserved 01 Nibble mode (MII/RMII 10/100bps) 10 Byte mode.Note: Reserved for 832x. 11 Reserved
24	SRP	Soft Receive Preamble 0 - Do not place the Receive Preamble before the Data 1 - Place eight bytes of the received Preamble and SFD before the data in the Data Buffer pointed by the first buffer descriptor of the frame. **Note:** In RMII this bit should be cleared

Table 29-10. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
25	STP	Soft Transmit Preamble 0 - Transmit normal preamble of length MACCFG2[PREL], and use the data in the data buffer as regular frame payload. 1 - If TxBD[PP]=1, use the first 8 bytes of the data buffer as 7 bytes of preamble and one byte of place holder.The Ethernet controller transmits the SFD automatically. All eight bytes must reside in one data buffer pointed by one Buffer Descriptor. If this bit is set and TxBD[PP]=0 the UEC transmits 7 bytes of preamble (compliant to 802.3). MACCFG2[PREL] must be programmed to 0x7 and the preamble length is 7 bytes. Other values of MACCFG2[PREL] result in undefined behavior of the UEC. **Note:** In RMII this bit should be cleared **Note:** MACCFG2[STP] must be set if TxBD[PP] is set.
26	Reserved	Must be set to ONE.
27	LC	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. If the actual received frame length does not match the value in the length field the value of InRangLenRxER statistics counter is incremented.
28	MPE	Magic Packet Enable 0 - The Magic Packet Detection Feature is disabled 1 - Magic Packet Detection is enabled. No frames are received by the UEC after the current frame has been received. The UEC automatically detects a Magic packet and as result asserts UCCE[MPD] and resets this bit. If unmasked an interrupt is issued to the CPU which wakes it up from low power mode.
29	PAD/CRC	Pad to 64 bytes 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	CRE	CRC Enable. If the configuration bit PAD/CRC ENABLE or the per-frame PAD/CRC ENABLE is set, CRC ENABLE 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	FDX	Full-duplex configure. This bit is cleared by default. 0 The MAC operates in half-duplex mode. 1 The MAC operates in full-duplex mode.

29.5.1.6 Interframe Gap/InterFrame Gap Register (IPGIFG)

IPGIFG is written by the user.

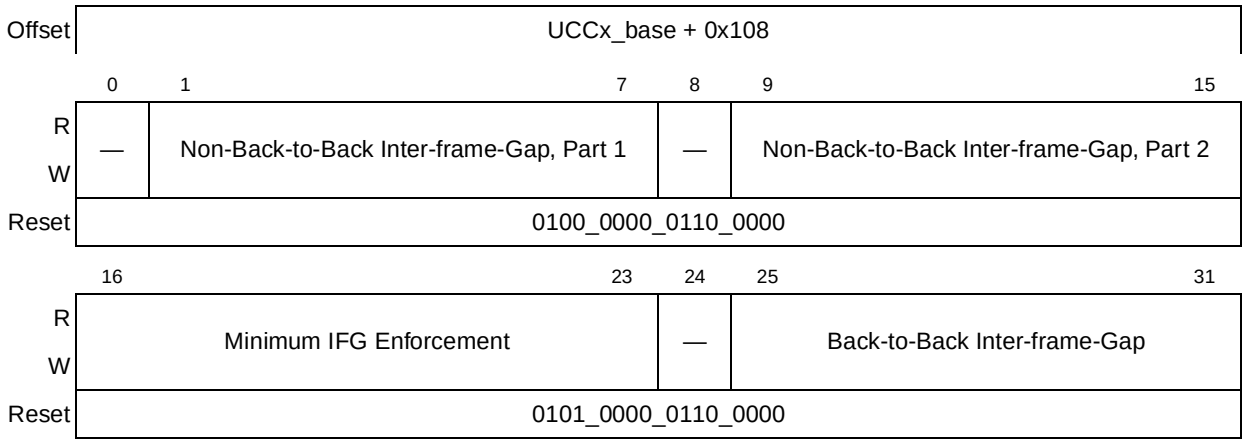


Figure 29-18. Interframe Gap/InterFrame Gap Register (IPGIFG)

Table 29-11 describes the IPGIFG fields.

Table 29-11. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-frame-Gap, Part 1	This is a 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, 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-frame-Gap, Part 2	This is a programmable field representing the non-back-to-back inter-frame-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	This is a 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-frame-Gap	This is a programmable field representing the IPG between back-to-back frames. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit frames 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.

29.5.1.7 Half-Duplex Register (HAFDUP)

HAFDUP is written by the user. Figure 29-19 describes the HAFDUP register.

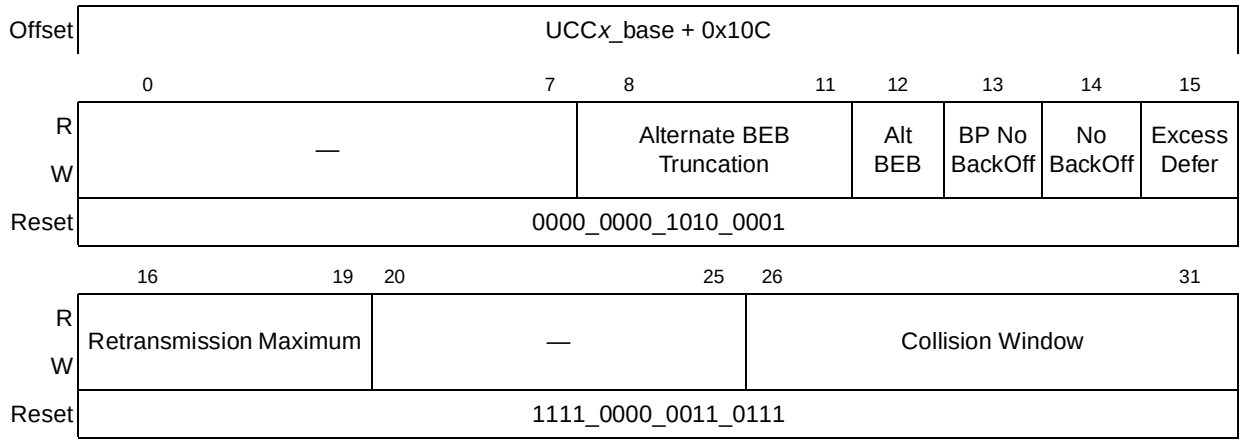


Figure 29-19. Half-Duplex Register (HAFDUP)

Table 29-12 describes the HAFDUP fields.

Table 29-12. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while ALTERNATE BINARY EXPONENTIAL BACKOFF 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 backoff. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential backoff rule. 1 The Tx MAC uses the ALTERNATE BINARY EXPONENTIAL BACKOFF 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 backoff time.
13	BP No BackOff	Back pressure no backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential backoff rule. 1 The Tx MAC immediately re-transmits, following a collision, during backpressure operation.
14	No BackOff	No backoff. 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	Setting this bit configures the Tx MAC to allow the transmission of a frame that is excessively deferred. An excess defer condition occurs when the Ethernet controller waits for more than 3036-byte time while attempting to send a frame. Clearing this bit causes the Tx MAC to abort the transmission of a frame that is excessively deferred. It is set by default.
16–19	Retransmission Maximum	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the frame due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d). Its default value is 0xF

Table 29-12. HAFDUP Field Descriptions (continued)

Bits	Name	Description
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.

29.5.1.8 MII Management Configuration Register (MIIMCFG)

MIIMCFG is written by the user. Figure 29-20 shows the MIIMCFG.

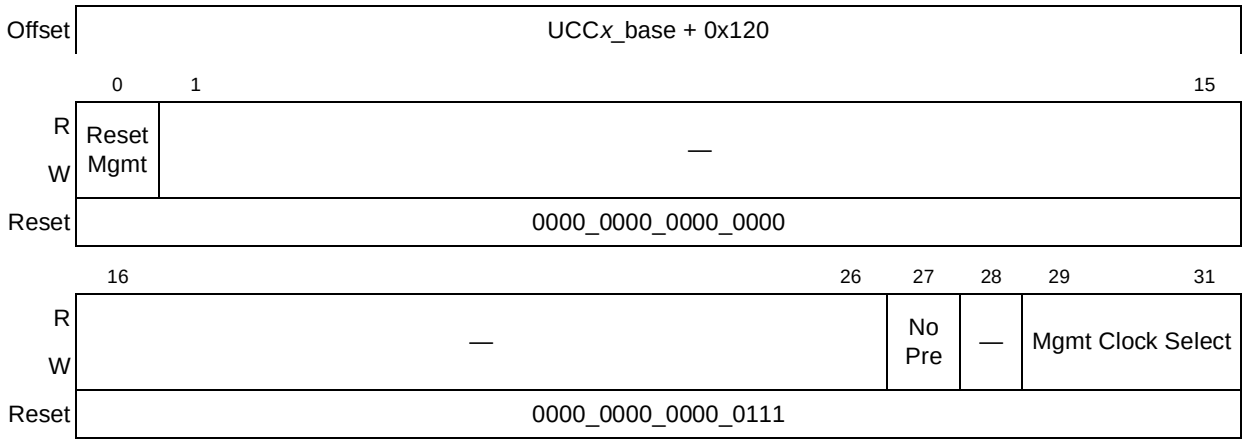


Figure 29-20. MII Management Configuration Register (MIIMCFG)

Table 29-13 describes the MIIMCFG fields.

Table 29-13. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII MGMT to perform mgmt 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 Mgmt read/write cycles with 32 clocks of preamble. 1 The MII MGMT suppresses preamble generation and reduces the Mgmt cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.

Table 29-13. MIIMCFG Field Descriptions (continued)

Bits	Name	Description
28	—	Reserved
29–31	Mgmt Clock Select	This field determines the clock frequency of the Mgmt Clock (CE_MDC). Its default value is 111. The source clock frequency is equal to the QUICC Engine clock divided by 16. 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

29.5.1.9 MII Management Command (MIIMCOM)

MIIMCOM is written by the user. Figure 29-21 shows the MIIMCOM register.

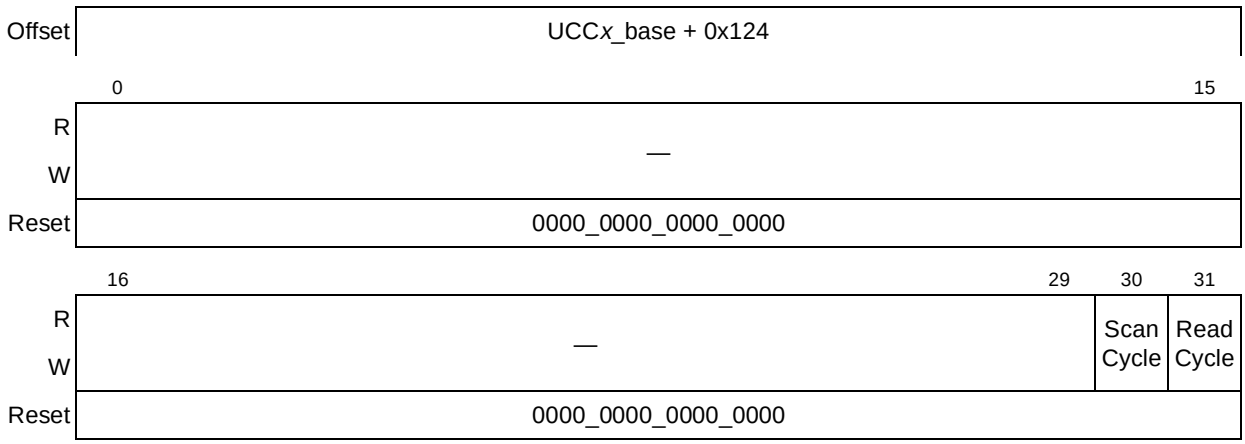


Figure 29-21. MII Management Command (MIIMCOM)

Table 29-14 describes the MIIMCOM register fields.

Table 29-14. MIIMCOM 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].
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].

29.5.1.10 MII Management Address Register (MIIMADD)

MIIMADD is written by the user. Figure 29-22 shows the MIIMADD register.

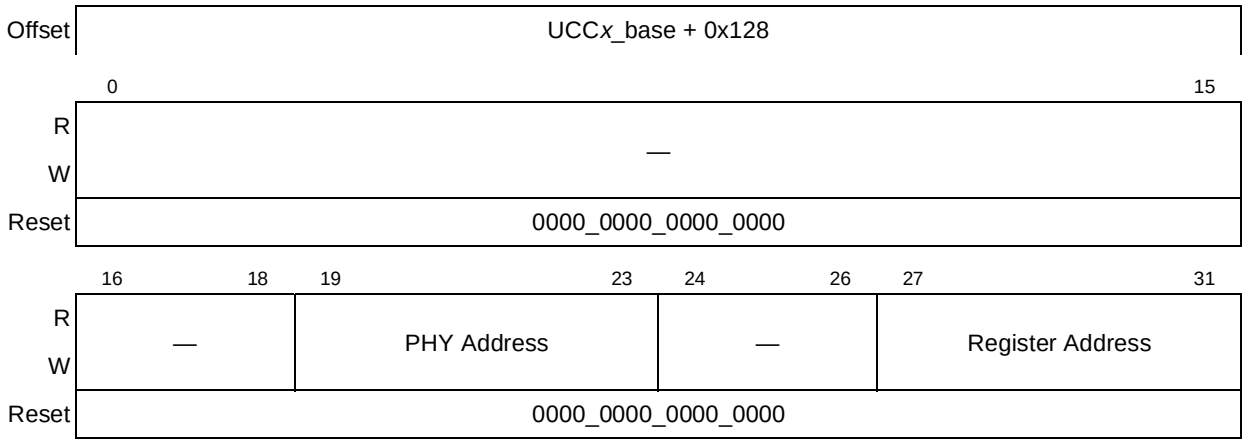


Figure 29-22. MII Management Address Register (MIIMADD)

Table 29-15 describes the MIIMADD fields.

Table 29-15. MIIMADD Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of Mgmt cycles. Up to 31 PHYs can be addressed, with one address reserved for internal TBI registers (corresponding to the value of TBIPA[TBIPA] whose default value is 0x00). Re-programming of TBIPA[TBIPA] to a non-zero value allows PHY address to then be set to 0.
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of Mgmt cycles. Up to 32 registers can be accessed. Its default value is 0x00.

29.5.1.11 MII Management Control Register (MIIMCON)

MIIMCON is written by the user. Figure 29-23 shows the MIIMCON register.

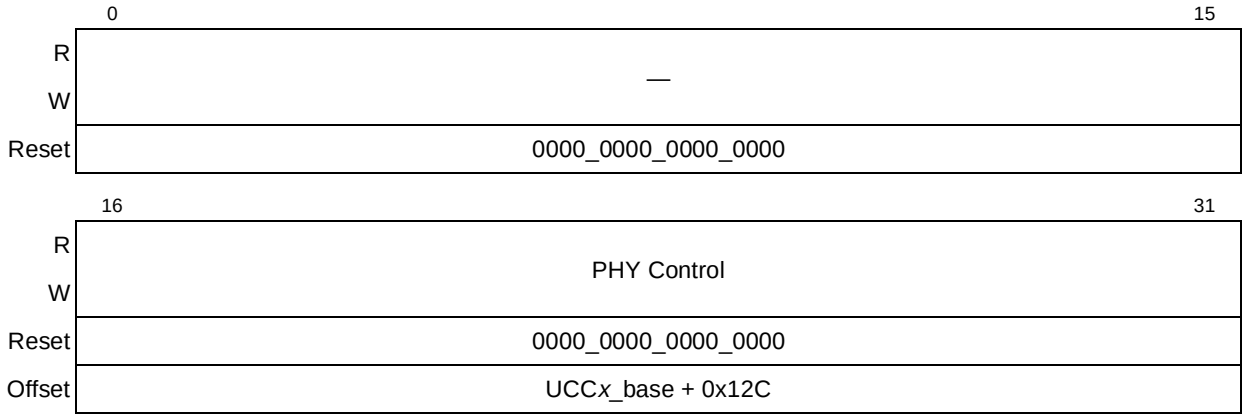


Figure 29-23. MII Management Control Register (MIIMCON)

Table 29-16 describes the MIICON fields.

Table 29-16. 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.

29.5.1.12 MII Management Status Register (MIIMSTAT)

MIIMSTAT is read-only by the user. Figure 29-24 shows the MIIMSTAT register.

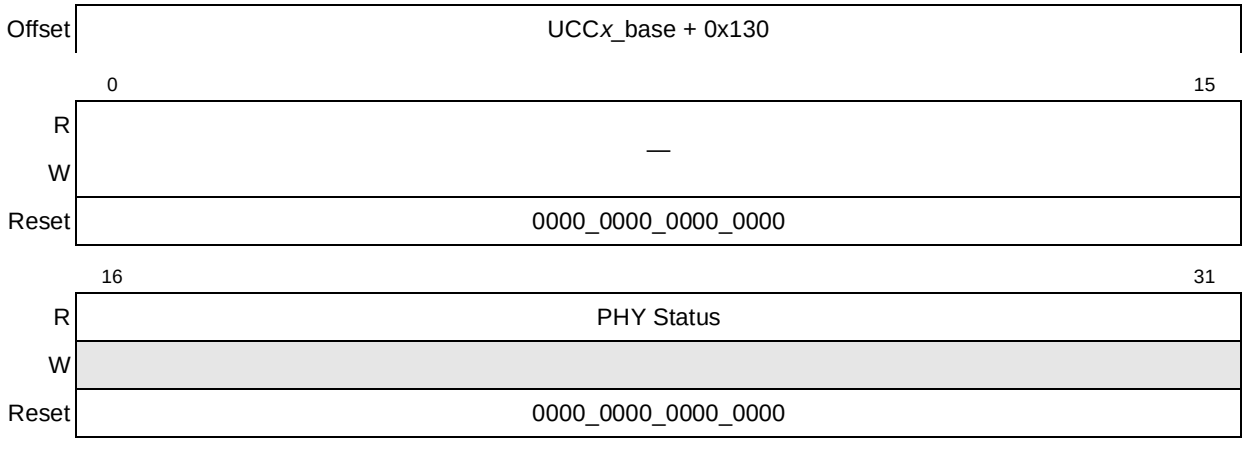


Figure 29-24. MII Management Status Register (MIIMSTAT)

Table 29-17 describes the MIIMSTAT fields.

Table 29-17. 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.

29.5.1.13 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 29-25 describes the definition for the MIIMIND register.

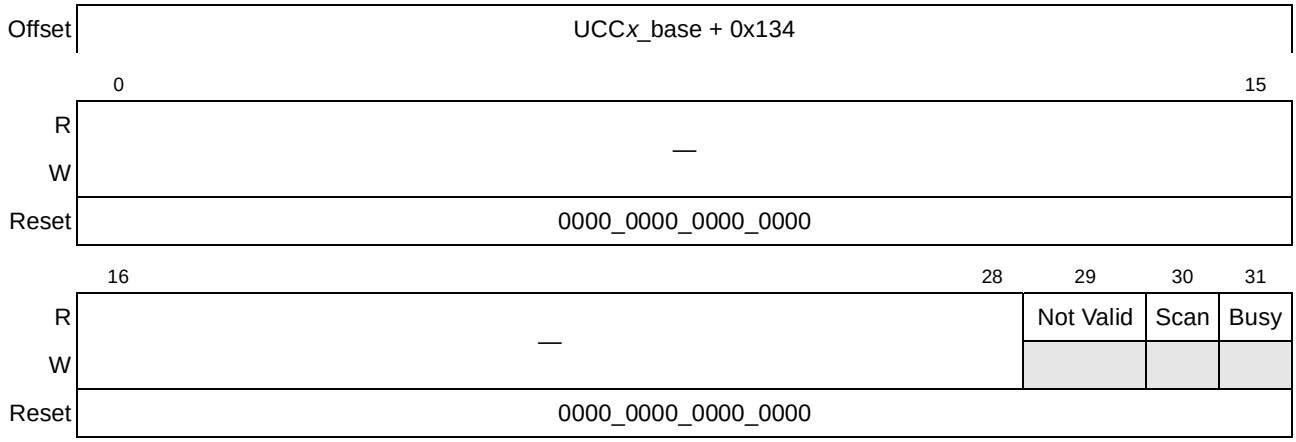


Figure 29-25. MII Management Indicator Register (MIIMIND)

Table 29-18 describes the MIIMIND fields.

Table 29-18. 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.
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.

29.5.1.14 Interface Status Register (IFSTAT)

IFSTAT is read by the user. Figure 29-26 shows the IFSTAT register.

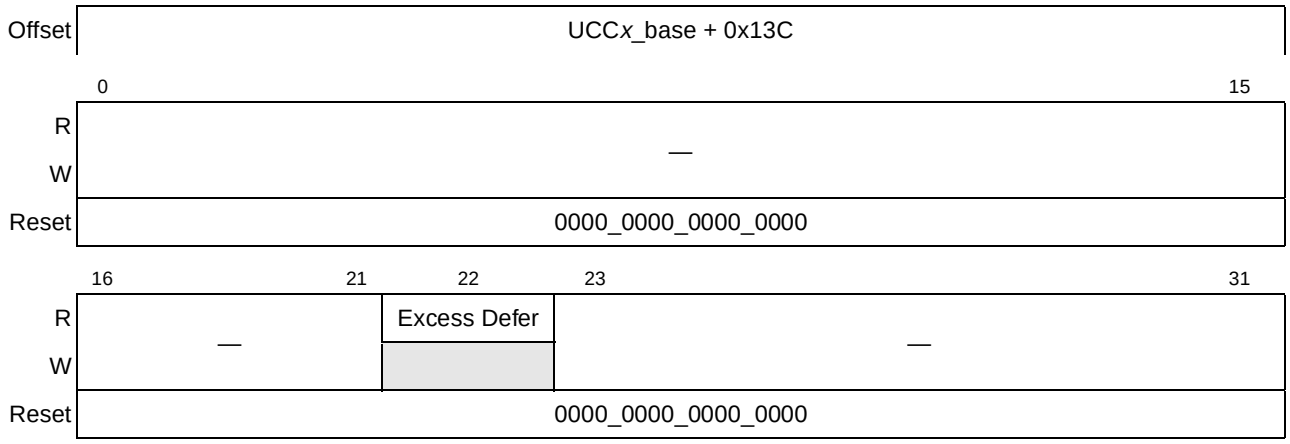


Figure 29-26. Interface Status Register (IFSTAT)

Table 29-19 describes the IFSTAT fields.

Table 29-19. 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

29.5.1.15 Station Address Part 1 Register (MACSTNADDR1)

MACSTNADDR1 register combined with MACSTNADDR2 are used for two purposes:

- Address filtering
- Source address for Flow Control frames

The MACSTNADDR1 register is written by the user. Figure 29-27 describes the definition for 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 will be returned. A read of MACSTNADDR2 will return 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.

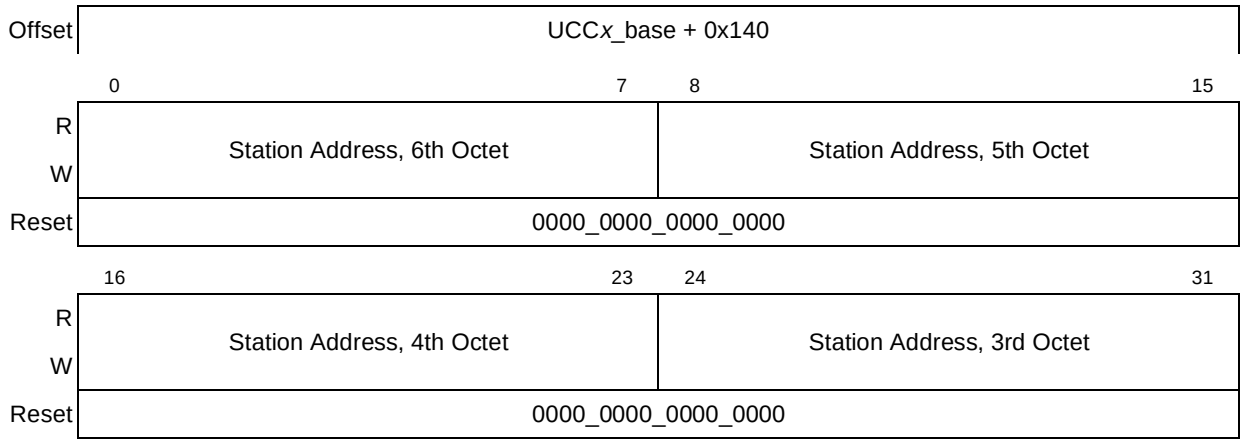


Figure 29-27. Station Address Part 1 Register Definition

Table 29-20 describes the MACSTNADR1 fields.

Table 29-20. MACSTNADR1 Field Descriptions

Bits	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet is stored in 40–47and 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 is stored in 32–39 and 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 is stored in 24–31 and 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 is stored in 16–23 and defaults to a value of 0x00.

29.5.1.16 Station Address Part 2 Register (MACSTNADDR2)

MACSTNADDR2 is written by the user. Figure 29-28 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.

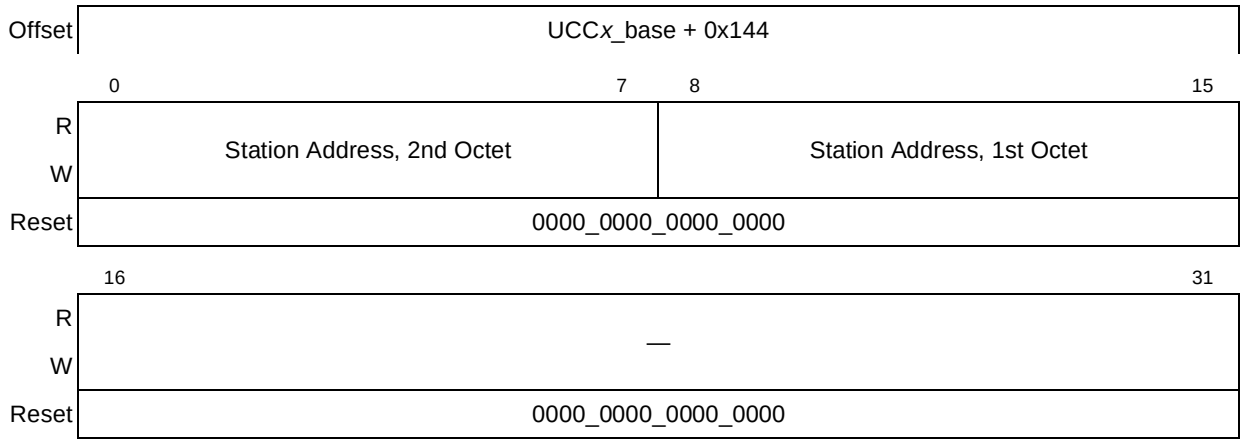


Figure 29-28. Station Address Part 2 Register (MACSTNADDR2)

Table 29-21 describes the MACSTNADDR2 fields.

Table 29-21. 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 is stored in 8–15 and 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 is stored in 0–7 and defaults to a value of 0x00.
16–31	—	Reserved

29.5.1.17 UCC Ethernet Mac Parameter Register (UEMPR)

offset	UCC*x*_base + 0x150															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Pause time value (MAC Parameter)															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	Extension Header															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-29. UCC Ethernet Mac Parameter Register (UEMPR)

Table 29-22. UEMPR Field Descriptions

Bits	Name	Description
0–15	PT	Pause time value (MAC Parameter). The quanta is 512 bits time.
16–31	PTE	Extension Header. See details on page 29-11.

29.5.1.18 UCC Ethernet Statistics Control Register (UESCR)

Offset	UCC*x*_base + 0x158															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	AUTOZ	CLRCNT	MAXCOV						SCOV							
R/W	R/W															
Reset	0000_1000_0000_0100															

Figure 29-30. Ethernet Event/Mask Register (UESCR)

Table 29-23 describes the UESCR fields.

Table 29-23. UESCR Field Descriptions

Bits	Name	Description
0	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.
1	CLRCNT	Clear all statistics counters 0 Allow MSTAT counters to continue to increment. 1 Reset all MSTAT counters. This bit is self-resetting.
2–7	MAXCOV	Maximum Coalescing Value. This field is used to limit the latency of the uCode in returning Tx buffers to the CPU. The maximum latency is SCOV * MAXCOV * “time to transmit 64 bytes” Default value after reset is 8. The user should not alter the value of this field.
8–15	SCOV	Status Coalescing Value. 0x1-0xFF - The default value after reset is 0x4. This field impacts the performance of the QUICC Engine. It does not affect the functional operation of the QUICC Engine block. The user should not alter the value of this field. 0x0 value is not legal.

29.5.2 Buffer Descriptors

A buffer descriptor is a fixed-size, aligned block of memory that points to a buffer of data for UEC to handle. The UCC Ethernet controller attempts to prefetch the Buffer Descriptors in groups of four. Certain restriction apply to the BD ring as described in the following sections.

29.5.2.1 Transmit Data Buffer Descriptor (TxBD)

Data is presented to the UCC Ethernet controller for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The following restrictions apply to the transmit BD Ring:

- The minimum BD ring size (such as number of BDs in the transmit BD ring) must be at least two TxBDs.
- The BD ring must be aligned to 32byte boundary.
- The user must allocate memory size of multiple of 32 bytes for the BD ring. If the size of the BD ring is not a multiple of 32 bytes, the user must pad the remaining bytes with zeroes.
- In VLAN insertion mode other restriction apply. See Figure 29-31 TxBD[VID] field.
- The BD ring size in units of BDs must be greater than the maximum number of TxBDs that construct a frame (such as, it is forbidden to occupy an entire TxBD ring for a single frame).
- At least one of the following restrictions must be met:
  - When multiple TxBDs are associated with a single frame, the user must set the Rbit of the first TxBD only when the entire frame and the other BDs have been placed in the external memory,

and all Rbits of all BDs are set (such as the Rbit of the first BD in a Tx frame must be set last by the CPU).

— The TxBD Ring size must be larger than (UESCR[SCOV] * M + 1) where M=Maximum number of BDs per frame plus one. UESCR[SCOV] default value is four.

The following suggestions yield to better performance

- Single BD per frame.
- The user should set the Rbit in the BDs in groups of four. The UEC prefetches four BDs at one time. The UEC fetches the BDs with Rbit cleared again and again until it finds the Rbit set.
- The prefetching mechanism for the TxBDs is not optimal if the size of the TxBD ring is smaller than four TxBDs.

The Ethernet controller 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 RMON block.

Figure 29-31. Transmit Buffer Descriptor

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**R**	**PAD/ CRC**	**W**	**I**	**L**	**TC**	DEF	**PP/** ExDef	LC	**IPCHI0 /RL**	RC/VID				**IPCHI1** /UN	**IPCHI2 /CSL**
Offset + 2	DATA LENGTH															
Offset + 4	TX DATA BUFFER POINTER															
Offset + 6																

Table 29-24. Transmit Buffer Descriptor Field Description

Offset	Bits	Name	Description
Offset + 0	0	R	Ready, written by UCC Ethernet controller and user. If TEMODER[IRT]=0 this bit is interpreted as follows: 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 UCC Ethernet controller 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. If TEMODER[IRT]=1 the polarity of this bit is reversed.
Offset + 0	1	PAD/CRC	PAD/CRC. Padding and CRC attachment for frames. 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, UCC Ethernet controller PADs always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames. **Notes:** 1. Valid only if MACCFG2[PAD/CRC enable] is cleared. If MACCFG2[PAD/CRC enable] is set, this bit is ignored. 2. This bit must be set in the last TxBD of the frame. It is ignored if set in other TxBDs of the frame.

Table 29-24. Transmit Buffer Descriptor Field Description (continued)

Offset	Bits	Name	Description
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 UCCE[TXB] or UCCE[TXE] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, UCCM[TXBEN] or UCCM[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. 0 End transmission immediately after the last data byte with no hardware generated CRC appended. 1 Transmit the CRC sequence after the last data byte. **Notes:** 1. Valid only if MACCFG2[PAD/CRC enable] is cleared. If MACCFG2[PAD/CRC enable] is set, this bit is ignored. 2. This bit must be set in the last TxBD of the frame. It is ignored if set in other TxBDs of the frame.
Offset + 0	6	DEF	Hardware updates this bit when an excess defer condition occurs.DEF—Defer indication, written by hardware. 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	PP/ExDef	Programmable Preamble (programmed by the CPU) 0 - Do not transmit a programmable preamble. 1 - Transmit Programmable preamble. The size of the preamble programmed in register MACCFG2[PREL] field must be 0x7; the preamble bytes are prepended to the data in the data buffer. Excessive Defer (written by the UEC) 0 - No defer 1 - Excessive Defer has occurred in this frame. **Note:** MACCFG2[STP] must be set if TxBD[PP] is set. **Note:** This bit must be set in the first TxBD of the frame. It is ignored if set in other TxBDs of the frame.
Offset + 0	8	LC	Late collision (written by UCC Ethernet controller). 0 No late collision. 1 A collision occurred after 64 bytes are sent. The UCC Ethernet controller terminates the transmission and updates LC.

Table 29-24. Transmit Buffer Descriptor Field Description (continued)

Offset	Bits	Name	Description
Offset + 0	9	IPCH0/ RL	IPCHI0, IPCHI1 (bit 14), IPCHI2 (bit15). Written by the CPU. Number of entry in IPH_Offset in Section 29.5.3.3, “Tx Global Parameter RAM” used as an offset from the beginning of the frame from which IP checksum is calculated for the transmit frame. If the table contains a 0xFF the checksum is not calculated. These bits must be set in the first TxBD of the frame. 000 - Use entry number 0 001 - Use entry number 1 ..... 111 - Use entry number 7 Note that the IP header must reside in the first TxBD of the frame, and in the first 128 bytes of the frame. The IPCH[0..2] bits must be programmed to a valid value in the first BD of the frame. In other TxBDs of the frame this field is ignored. The value of the IP checksum field in the frame in memory must be zero. Retransmission Limit. Written by UCC Ethernet controller. 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 UCC Ethernet controller terminates the transmission and updates RL.
Offset + 0	10–13	RC / VID	Retry Count. Written by UCC Ethernet controller. 0 The frame is sent correctly the first time or if RL is set then the retry limit has been reached 1 One or more attempts where 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. VLAN Tag ID and Enable; Written by the Host CPU. ID of VLAN Tag to be transmitted. The VLAN Tag value resides in a Table located in the UCC Global multiport RAM at VTAGTable. 0xxx - Do not Insert a VLAN Tag on the Transmitted Frame 1nnn- Use VLAN Tag at index VTAGTable+0xnnn For VLAN Tag ID insertion, this field must be programmed to ‘1nnn’ only in the first BD of the frame. Other Tx BDs in the frame must have this field programmed to ‘0000’. The Data length of the first TxBD of the frame must be >= 12 bytes.
Offset + 0	14	IPCHI1/ UN	IPCHI1. See description for bit 9. Underrun. Written by UCC Ethernet controller. 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 UCC Ethernet controller terminates the transmission and updates UN.
Offset + 0	15	IPCHI2/ CSL	IPCHI2. See description for bit 9. Carrier sense lost. Carrier sense is lost during frame transmission. The Ethernet controller updates CSL after sending the buffer.

Table 29-24. Transmit Buffer Descriptor Field Description (continued)

Offset	Bits	Name	Description
Offset + 2	0–15	Data Length	Data length is the number of octets the UCC Ethernet controller should transmit from this BD's data buffer. It is never modified by the UCC Ethernet controller. This field must be greater than zero. 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.

29.5.2.2 Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E, I, and W bits in the first word and the pointer in the second word. The first word of the RxBD contains control and status bits. The UCC Ethernet controller modifies the E, L, F, M, BC, MC, LG, NO, SH, CR, OV, and IPCH 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 IPCH bits in the first word of the buffer descriptor are valid if the L bit is set.

The following restrictions apply to the receive BD Ring:

- The BD ring size (such as number of BDs in the receive BD ring) must be modulo four.
- The smallest numbers of BDs in a BD ring is eight
- The BD ring must be initialized with empty BDs (E=1) before the Ethernet init command is executed. Otherwise a busy condition occurs on the first frame
- The BD ring must be aligned to 32byte boundary

The following suggestions yield to better performance

- The user should set the E bit in the BDs in groups of four. The UEC prefetches four BDs at one time. The UEC fetches the BDs with E bit cleared again and again until it finds the E bit set.
- It is suggested to initialize a BD ring that can host multiple frames to avoid busy condition
- Single BD per frame
- The minimum buffer size (MRBLR) is 128 bytes. This is the Virtual FIFO block size programmed for the Ethernet protocol. See Section 26.2, "UCC Virtual FIFOs" for more details on Virtual FIFO.
- MRBLR must be a multiple of Virtual FIFO block size (such as a multiple of 128 bytes)

In the RxBD, the user initializes the E, I, and W bits in the first word and the pointer in the second word. If the data buffer is used, UCC Ethernet controller 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 IPCH bits in the first word of the buffer descriptor are only modified by UCC Ethernet controller if the L bit is set. The first word of the RxBD contains control and status bits. Its format is detailed below. Figure 29-32 describes the definition for the RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	Res	W	I	L	F	Res	M	BC	MC	LG	NO	SH	CR	OV	IPCH
Offset + 2	DATA LENGTH															
Offset + 4	RX DATA BUFFER POINTER															
Offset + 6																

Figure 29-32. Receive Buffer Descriptor

Table 29-25. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty, written by the UCC Ethernet controller (when cleared) and by the 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	—	Reserved. Set to zero.
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 UCCE[RXB0..7] is set after this buffer is serviced. This bit can cause an interrupt if enabled (UCCM[RXBEN1..7]).
Offset + 0	4	L	Last in frame, written by the UCC Ethernet controller. 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 the UCC Ethernet controller. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
Offset + 0	6	Reserved	Set to zero.
Offset + 0	7	M	Miss, written by the UCC Ethernet controller. (This bit is valid only if the L-bit is set and UCC Ethernet controller is in promiscuous mode and if REMODER[EXP]=0) This bit is set by the UCC Ethernet controller 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. If REMODER[EXP]=1. Valid only when L bit is set in the RxBD This bit is set is LAST_PCD[BDM1] is set.
Offset + 0	8	BC	Broadcast. Written by UCC Ethernet controller. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).

Table 29-25. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	9	MC	Multicast. Written by UCC Ethernet controller (Only valid if L is set.) Note that if BC is set, this bit is also set.
Offset + 0	10	LG	Rx frame length violation, written by the UCC Ethernet controller (only valid if L is set). A frame length greater than maximum frame length was recognized. This bit is valid only if the L-bit is set.
Offset + 0	11	NO	Rx non-octet aligned frame, written by the UCC Ethernet controller (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 the UCC Ethernet controller (only valid if L is set). A frame length that was less than the minimum length defined for this channel (MINFLR) was recognized,.
Offset + 0	13	CR	Rx CRC error, written by the UCC Ethernet controller (only valid if L is set). The CR bit in the BD is set in the following cases: a) The CRC value a the end of the frame does not match the CRC calculated on the frame received from the line before any header modification. b) If a receive code group error is detected, or c) The previous frame got rx_er (code symbol error) and in addition it has been dropped due to an IPG violation (IPG smaller then 96 bit time).
Offset + 0	14	OV	Overrun, written by the UCC Ethernet controller (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, CR lose their normal meaning and are zero and should be ignored.
Offset + 0	15	IPCH	IP Checksum written by the UCC Ethernet controller if UPSMR[IPCHK]=1) 0 IP checksum check on IPv4 header passed 1 IP checksum check on IPv4 header failed Note: This bit is NOT changed by the UCC Ethernet controller if UPSMR[IPCHK]=0.
Offset + 2	0–15	Data Length	Data length, written by the UCC Ethernet controller. Data length is the number of octets written by the UCC Ethernet controller 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. Note that when IPAE is enabled (IP address alignment) the frame length does not include the two extra 'garbage' bytes inserted at the beginning for purpose of alignment.
Offset + 4	0–31	Rx Data Buffer Pointer	Receive buffer pointer, written by the 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 UCC Ethernet controller.

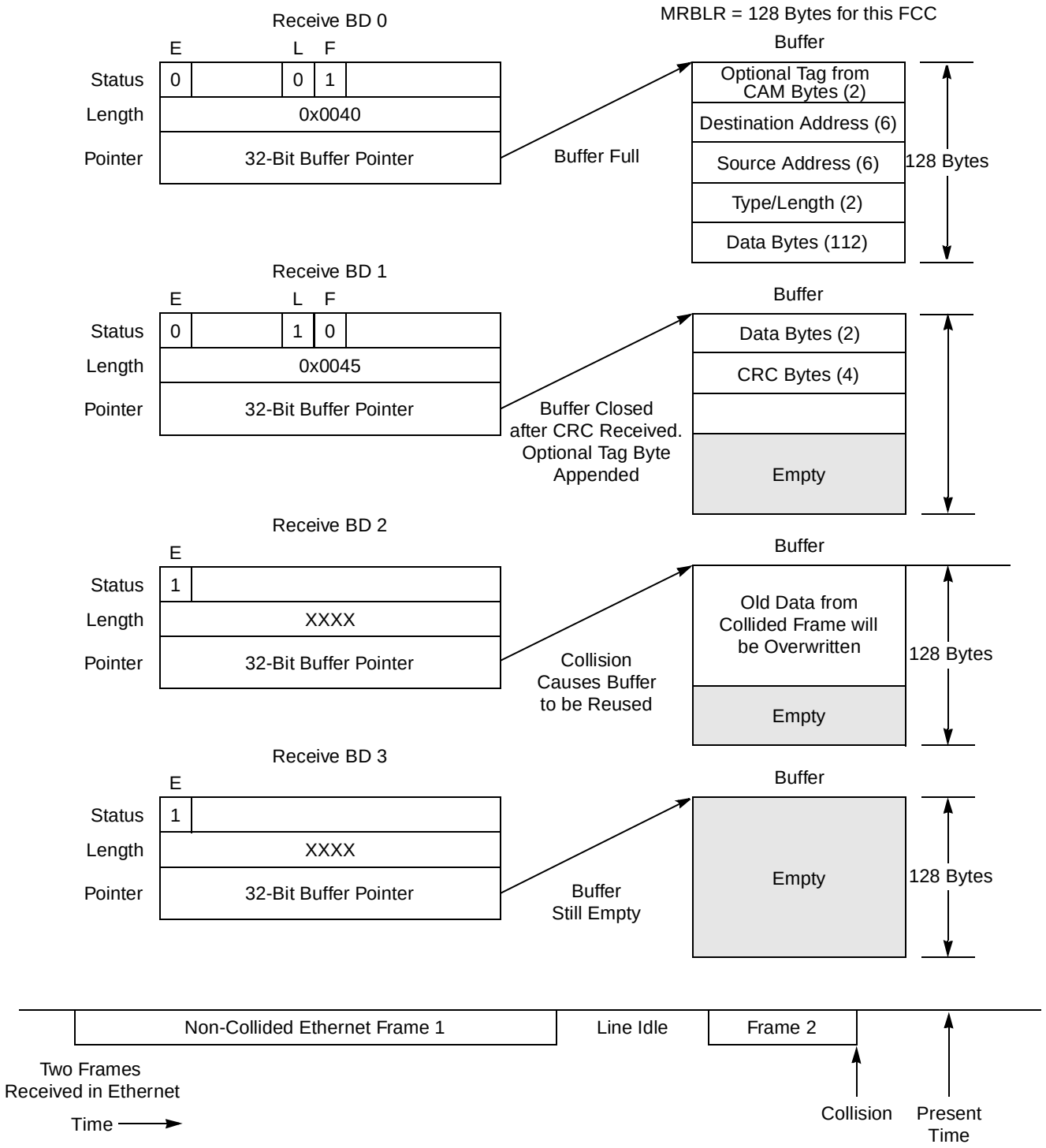


Figure 29-33. Example of Ethernet Receiving Using RxBD

29.5.3 Parameter RAM

The UCC Ethernet transmitter and receiver have their own individual Parameter RAM. Each Parameter RAM is split into two types: Global Parameter RAMs and Thread Parameter RAM.

The BASE Address for the UCC Parameter RAM and of Thread Parameter RAM is determined by the QUICC Engine Commands Section 29.8.1, “Init Tx, Init Rx and InitTx and Rx Parameters Command.” The number of threads that each UCC runs is configured in the same command.

The Ethernet INIT command must be executed after the Rx and Tx Parameter RAM have been initialized by the user.

29.5.3.1 Transmitter Parameter RAM Overview

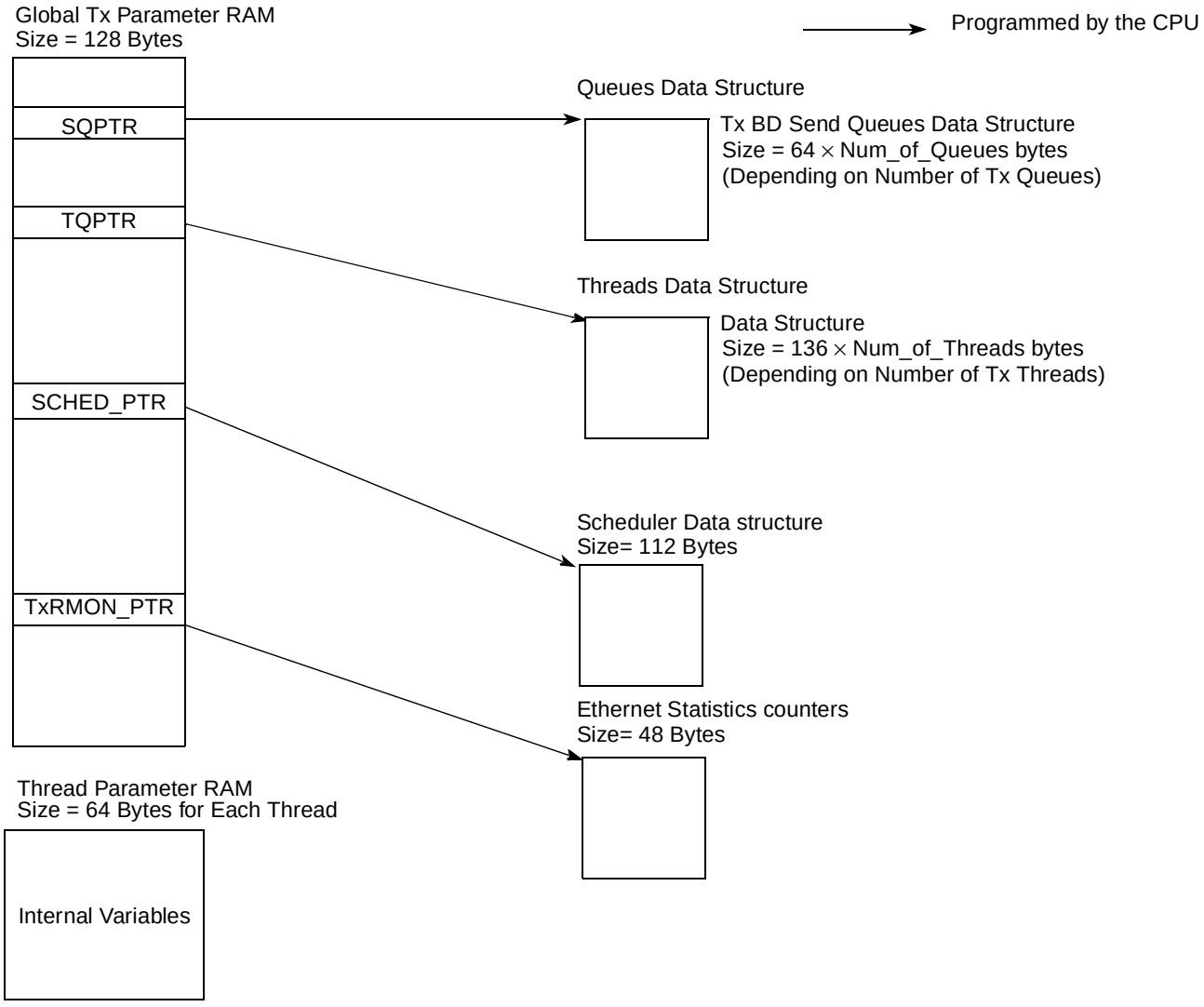


Figure 29-34. Transmitter Parameter RAM Data Structures

29.5.3.2 Receiver Parameter RAM Overview

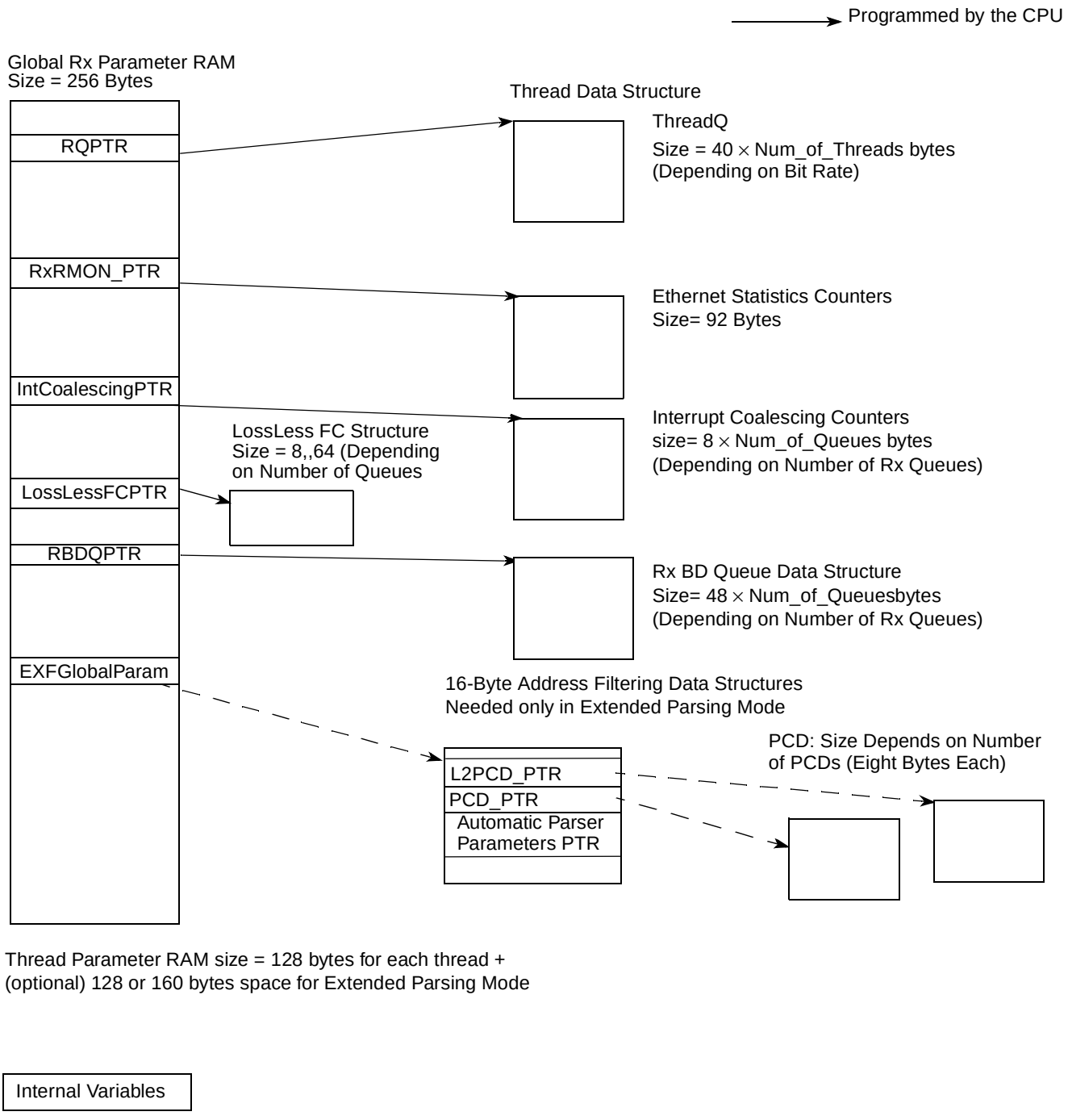


Figure 29-35. Receiver Parameter RAM data structures for Termination

29.5.3.3 Tx Global Parameter RAM

Table 29-26 describes the Tx Global Parameter RAM

Table 29-26. Tx Global Parameter RAM

Offset	Bits	Name	Description	Initialized by
0x0–0x1 TEMODER	—	TEMODER	Transmit Ethernet Mode Register.	CPU
	0	StartOfFrame	Internal variable. Initialize to 1.	CPU
	1	StartOfBD	Internal variable. Initialize to 1.	CPU
	2	SchedEnable	When set the frame scheduler is enabled otherwise the scheduler is disabled. When the number of send queues is 1 the scheduler shall be disabled.	CPU
	3	PrefAndContFlag	Internal variable.	CPU
	4	Reserved	Internal variable.	CPU
	5	IPCHK	Enable IP Checksum Generate 0 - Do not generate IP checksum in IPv4 frames 1 - Generate IP checksum in IPv4 frames Note: The value of the IP checksum in the IP header placed by the CPU in memory must be zero Note: The IP header must reside in the first TxBD of the frame, and in the first 128 bytes of the frame.	CPU
	6	Reserved	Internal variable.	CPU
	7	RMONEN	When set RMON statistics counters are enabled.	CPU
	8–9	Res	Set to zero	CPU
	10–12	Reserved	Reserved. Initialize to zero.	CPU
	13–15	NumOfQueues	Number of Tx queues 000 - One Tx queue 001 - Two Tx queues ... 111 - Eight Tx queues	CPU
0x2–0x37	XXX bytes	Reserved	—	QE
0x38–0x3B	0–15	SQPTR	Base address of the send queue memory region. See Section 29.5.3.3.2, “Tx Send Queue Memory Region.” This address must be 32 bytes aligned.	CPU
	0–15			
0x3C–0x3F	0–15	SchedulerBasePointer	Base address of the scheduler memory region (SCBASE). See Section 29.5.3.3.3, “Scheduler Programming Model and Data Structures.”	CPU
	0–15			
0x40–0x43	0–15	TxRMONBasePointer	Base address of the Tx RMON statistic counter. See Section 29.7.4.1, “TX Firmware Counters.”	CPU
	0–15			

Table 29-26. Tx Global Parameter RAM (continued)

Offset	Bits	Name	Description	Initialized by
0x44–0x47	0–31	TSTATE	Transmit internal state. The high byte of TSTATE, bit 0 to 7, contains the Bus Mode Register which is programmed by the Host CPU at initialization. Other bits in this register are for internal QUICC Engine block usage. See Section 29.5.4, "Bus Mode Register (BMRx)." The rest of the bits are initialized to zero.	CPU
0x48–4F	0–31	IPH_Offset0,1...7	Offset (in bytes) from the beginning of the packet to the beginning of the IP header. IPH_Offset0,1...7 value is used if TxBD[IPCHI0,IPCHI1,IPCH12] = 000,001...111 respectively. The following restrictions apply: 1. The IP header offset is the offset from the TxBD[Tx Data Buffer Pointer] 2. Minimum value is 0 maximum value is 128-size of IP header. 3. 0xFF means that no checksum is calculated	CPU
0x50–0x6f	0–15 0–15	VTAGTable	VLAN Tag Table. Consists of up to 8 4-byte VLAN tags. The table must be initialized after reset if VLAN insertion is employed.	CPU
0x70–0x73	8–31	TQPTR	Base Address of the Tx Queues Memory Region. This address must be aligned to: 64 bytes in case of one thread is used. 128 bytes in case of four threads are used.	CPU
0x74–0x77	—	Reserved	Internal variable.	N/A

29.5.3.3.1 Thread Data Structure

This data structure is located at base address TQPTR. The data structure is initiated by the QUICC Engine block upon issuing an INIT Tx Ethernet Command. The user must allocate 136 bytes per thread.

29.5.3.3.2 Tx Send Queue Memory Region

The send queue memory region is pointed by the SQPTR in the Tx Global Parameter RAM, consists of queue descriptor per every send queue (SQQD). Up to 8 send queues are supported. SQQDs are placed in a consecutive order in the Send Queue Memory region as illustrated by Table 29-27.

Table 29-27. Send Queue Descriptors

Offset	Bits	Name	Description
0x00–0x3F	—	SQQD0	Send queue 0 queue descriptor.
0x40–0x7F	—	SQQD1	Send queue 1 queue descriptor.
0x80–0xBF	—	SQQD2	Send queue 2queue descriptor.
0xC0–0xFF	—	SQQD3	Send queue 3 queue descriptor.
0x100–0x13F	—	SQQD0	Send queue 4 queue descriptor.
0x140–0x17F	—	SQQD1	Send queue 5 queue descriptor.
0x180–0x1BF	—	SQQD2	Send queue 6 queue descriptor.
0x1C0–0x1FF	—	SQQD3	Send queue 7 queue descriptor.

Every SQQD is 64 bytes long and is described in detail by Table 29-27.

Table 29-28. Tx Send Queue Descriptor (SQQD)

Offset	Size	Name	Description (Data structure must be 32 bytes aligned).	Initialized by
0x00–0x03	Word	BDRingBase	A pointer to the BD ring base address. This field must be initialized to the base address of the BD ring.	CPU
0x04–0x09	—	Reserved	Set to zero	QE
0x0C – 0x0D	—	LastBDCompleted Address	This field must be initialized after reset to point to the last entry in the BD ring.	CPU
0x0E–0x3F	4 Words	Reserved	Set to zero	QE

29.5.3.3.3 Scheduler Programming Model and Data Structures

Table 29-29 summarizes the programming model and data structures. The base address to the scheduler memory region is specified in the Tx Global parameter RAM.

Table 29-29. Scheduler Programming Model

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
0x0	CPUCount0	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #0	QE
0x2	CPUCount1	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #1	
0x4	CECount0	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value.	
0x6	CECount1	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value.	
0x8	CPUCount2	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #2	
0xA	CPUCount3	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #3	
0xC	CECount2	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value.	
0xE	CECount3	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value.	
0x10	CPUCount4	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #4	
0x12	CPUCount5	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #5	
0x14	CECount4	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value.	

Table 29-29. Scheduler Programming Model (continued)

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
0x16	CECount5	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value	QE
0x18	CPUCount6	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #6	
0x1A	CPUCount7	16	CPU Packet counter. CPU must increment this counter by one every time a frame is placed in TxBD ring #7	
0x1C	CECount6	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value	
0x1E	CECount7	16	QUICC Engine Packet counter. This entry is updated by the QUICC Engine block. The CPU must no alter its value	
0x20	WeightStatus0	32	Accumulated Weight factor for queue #0. Initialized by SW to 0	
0x24	WeightStatus1	32	Accumulated Weight factor for queue #1. Initialized by SW to 0	
0x28	WeightStatus2	32	Accumulated Weight factor for queue #2. Initialized by SW to 0	
0x2C	WeightStatus3	32	Accumulated Weight factor for queue #3. Initialized by SW to 0	
0x30	WeightStatus4	32	Accumulated Weight factor for queue #4. Initialized by SW to 0	
0x34	WeightStatus5	32	Accumulated Weight factor for queue #5. Initialized by SW to 0	
0x38	WeightStatus6	32	Accumulated Weight factor for queue #6. Initialized by SW to 0	
0x3C	WeightStatus7	32	Accumulated Weight factor for queue #7. Initialized by SW to 0	
0x40	RTSRShadow	32	Temporary variable handled by the QUICC Engine block	N/A
0x44	Time	32	Temporary variable handled by the QUICC Engine block	N/A
0x48	TTL	32	Temporary variable handled by the QUICC Engine block	N/A
0x4C	MBLInterval	32	Max Burst Length interval. This parameter determines the longest 1Gbps burst allowed by the Traffic Shaper. The user should program this parameter according to the following formula: MBLInterval = INTEGER[MBL * TSRByteTime] Where MBL is the maximum burst size in bytes allows by the receiver side, and TSRByteTime is the transmission time for one data byte expressed in TSR count units.	CPU
0x50	NorTSRByteTime	16	Contains the normalized value of Byte time in TSR units in the desired frequency. Calculated according to the formula: ROUND[TSRByteTime * 2^FracSiz]. See above for further explanation.	CPU

Table 29-29. Scheduler Programming Model (continued)

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
0x52	FracSiz	8	This parameter contains the radix 2 log value of the denominator of NorTSRByteTime, as follows: TSRByteTime = NorTSRByteTime/2^FracSiz See example calculations in Section 29.15, "Traffic Shaper Programming Considerations." Note: CPU must write a complete byte when Initializing this value or changing it on the fly (3 upper bits must be zero) although only the 5 lowest bits are actually used to determine the FracSiz value (FracSize value is between 0 to 31).The QE may change the unused bits (3 upper bits) on run time.	CPU
0x53	SCHSTATR	8	Scheduler Status register. See Section 29.5.3.3.4, "Scheduler Status Register (SCHSTATR)."	CPU
0x54	StrictPriorityQ	8	Strict Priority Mask register. Bit i of this register, i=0..7, corresponds to output queue 0:7 respectively. If the bit is set, the output queue associated with this bit belongs to the Strict Priority group (SP Queue). If the bit is cleared, the queue belongs to the WFQ group of queues (WFQ Queue)	CPU
0x55	TxASAP	8	Transmit ASAP Register. Bit i of this register, i=0..7, corresponds to output queue 0:7 respectively. If the bit is set, the packet in the output queue associated with this bit is transmitted ASAP, without Weighting to the existence of the Traffic Scheduler limitations as maximum burst and average data rate	CPU
0x56	ExtraBW	8	Extra bandwidth register. Bit i of this register, i=0..7, corresponds to output queue 0:7 respectively. If the bit is set, the queue corresponds to this bit is not consuming BW from the budget allowed by the Traffic Shaper. Rather, its consumed BW is added to the general line Bandwidth.	CPU
0x57	OldWFQMask	8	Temporary variable. Initialized to 0.	QE
0x58	WeightFactor0	8	Weight Factor for queue #0. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #0	CPU
0x59	WeightFactor1	8	Weight Factor for queue #1. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #1	
0x5A	WeightFactor2	8	Weight Factor for queue #2. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #2	
0x5B	WeightFactor3	8	Weight Factor for queue #3. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #3	
0x5C	WeightFactor4	8	Weight Factor for queue #4. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #4	

Table 29-29. Scheduler Programming Model (continued)

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
0x5D	WeightFactor5	8	Weight Factor for queue #5. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #5	CPU
0x5E	WeightFactor6	8	Weight Factor for queue #6. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #6	
0x5F	WeightFactor7	8	Weight Factor for queue #7. Indicates the size of the Weight period following the transmission of data quantum of size 2^LogBlockLength bytes for Queue #7	
0x60	MinW	32	Temporary variable handled by QUICC Engine block. Initialized to 0.	QE
0x64	Reserved	32	Initialized to 0.	QE
0x68–0x6F	Reserved	—	Initialized to 0	QE

29.5.3.3.4 Scheduler Status Register (SCHSTATR)

Bits	0	1	2	3	4	5	6	7
Field	—	PAD/CRC	—			CRE	—	
R/W	R/W							
Reset	0000_0000_0000_0000							

Figure 29-36. Scheduler Status Register (SCHSTATR)

Table 29-30. SCHSTATR Field Descriptions

Bits	Name	Description
0	—	Reserved. Clear to zero
1	PAD/CRC	This bit is a reflection of MACCFG2[PAD/CRC] bit. this bit should be set to “1” if MACCFG2[PAD/CRC] bit is set and should be zero in case of MACCFG2[PAD/CRC] is zero.
2–4	—	Reserved. Clear to zero
5	CRE	This bit is a reflection of MACCFG2[CRE] bit. this bit should be set to “1” if MACCFG2[CRE] bit is set and should be zero in case of MACCFG2[CRE] is zero.
6–7	—	Reserved. Clear to zero

29.5.3.4 Tx Thread Parameter RAM

The TX thread parameter RAM is summarized by Table 29-31.

Table 29-31. Tx Thread Parameter RAM

Offset	Bits	Name	Description	Initialized by
0x00–0x3F	—	Reserved	Internal variable.	QE

29.5.3.5 Rx Global Parameter RAM

The RX Global Parameter RAM is summarized by Table 29-32.

Table 29-32. RX Global Parameter RAM

Offset	Bits	Name	Description	Initialized by
0x00	0–31	REMODER	Ethernet Mode Register. See Section 29.5.3.7, “Rx Ethernet Mode Register (REMODER).” Initialize to zero.	CPU
0x04	0–31	RQPTR	Base Address of the Rx Queues Memory Region. This address must be aligned to: 32 bytes in case of one thread is used. 128 bytes in case of four threads are used.	CPU
0x08–0x1F	0–31	Reserved	Initialize to zero	QE
0x20	—	Type_or_Len	If the value of Type/Len field in the frame is less than Type_or_Len value, then the Type/Len field in the frame indicates the length of the frame. If the value of Type/Len field in the frame is equal or greater than Type_or_Len value, then the Type/Len field in the frame indicates the protocol type of the next header in the frame.	CPU
0x22	0–14	Reserved	Set to zero	CPU
	15	RxGSTPAck	Receive Graceful STOP Host Command acknowledge 0 - Receive Graceful Stop Host Command is not completed by QR 1 - Receive Graceful Stop Host Command is completed, no more frames are received till Receive Restart Host Command is issued, or the UEC is reset and re-enabled. The CPU should reset this bit before issuing the Graceful Stop Host Command.	—
0x24	0–31	RxRMONBasePointer	Rx Ethernet Statistics Counters base address. This base address must be four bytes aligned.	CPU
0x28–0x2F	—	Reserved	Set to zero	QE
0x30	0–31	IntCoalescingPTR	Base Address to Interrupt Coalescing Table. The user must allocate a data structure of size (in bytes) = 8*number of Rx queues + 4. This base address must be 64 bytes aligned.	CPU
0x34	0–7	Busy_vector	Busy condition. one bit per queue. Set when a frame is received and discarded due to a lack of buffers. Initialize to zero.	N/A
0x36	0–7	RSTATE	Receive internal state. Reserved for QUICC Engine block use only. This byte contains the bus mode register. See Section 29.5.4, “Bus Mode Register (BMRx).”	CPU

Table 29-32. RX Global Parameter RAM (continued)

Offset	Bits	Name	Description	Initialized by
0x37–0x44	—	Reserved	Set to zero	QE
0x46	0–15	MRBLR	Maximum receive buffer length (a multiple of block size). The number of bytes that the UCC 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 UCC 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 UCC is operating. Change MRBLR only when the UCC receiver is disabled. MRBLR must be a multiple of the Virtual FIFO block size. See Section 29.5.2.2, “Receive Buffer Descriptor (RxBD)” for restriction on the buffer size. Note: If EMODER[DNE]=0 MINFLR<MRBLR. If EMODER[DNE]=1 MINFLR < (MRBLR-4)	CPU
0x48	0–31	RBDQPTR	RxBD Parameter Table Base Address. See Section 29.5.3.10, “RxBD Queue data structures”. For each Rx queue the user must allocate 16+32 bytes. This base must be eight byte aligned.	CPU
0x4C	0–15	MFLR	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 not discards the rest of that frame. The controller also reports the frame status and length of the received frame MFLR includes all in-frame bytes between the start frame delimiter and the end of the frame.	CPU
0x4E	0–15	MINFLR	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. MINFLR value must be smaller than MRBLR value. Note: If EMODER[DNE]=0 MINFLR<MRBLR. If EMODER[DNE]=1 MINFLR < (MRBLR-4)	CPU
0x50	0–15	MAXD1	Max DMA1 length register (typically1520 decimal)—Lets the user prevent system bus writes after a frame exceeds a certain 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 and reports the frame status and the frame length in the last RxBD. This value must be greater than block size. The frame length includes the discarded bytes.	CPU

Table 29-32. RX Global Parameter RAM (continued)

Offset	Bits	Name	Description	Initialized by
0x52	0–15	MAXD2	Max DMA2 length register (typically1520 decimal)— Lets the user prevent system bus writes after a frame exceeds a certain size. The value of MAXD2 is valid in promiscuous or Extended Parsing 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 and reports frame status and length 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 lower than MAXD1. The frame length includes the discarded bytes.	CPU
0x54	0–31	ECAM_PTR	RESERVED	CPU
0x58	0–31	L2QT	VLAN priority mapping table. See Section 29.5.3.11.1, “Rx Layer 2 QoS Table—L2QT.”	CPU
0x5C	32 Bytes	L3QT	IP priority mapping table. See Section 29.5.3.11.2, “Layer 3 QoS Table - L3QT.”	CPU
0x7C	0–15	VLAN_TYPE	VLAN Type. This field is inserted as part of the Q-TAG if enabled. Suggested value is 0x8100.	CPU
0x7E	0–15	TCI	Default TCI [VPri(3bits),CFI(1 bit),VID(12bits)]. This field is inserted or replaces the existing TCI as part of the Q-Tag if enabled.	CPU
0x80–0xBF	64 Bytes	Address Filtering (AF)	Address Filtering Data Structure. This structure definition depends on the address filtering mode: MPC82xx Compatible (REMODER[EXP]=0) or Extended Parsing mode (REMODER[EXP]=1). See Section 29.5.3.8, “Address Filtering (AF) Field Description” for more details.	CPU
0xC0	—	EXPGlobalParam	Base Address for Extended Parsing Global Parameters. See Section 29.6.1, “Extended Parsing Mode Global Parameters.” The user needs to allocate 16 bytes for this data structure. This base address must be eight bytes aligned.	CPU
0xC4	—	LossLessFCPtr	Base Address to Lossless Flow Control Data Structure. See Section 29.4.6.1.1, “Lossless Flow Control.”	—
0xF8–0xFF	—	Zero	Initialize to zero.	QE

29.5.3.6 Rx Thread Parameter RAM

The RX thread parameter RAM is summarized by Table 29-33.

Table 29-33. Rx Thread Parameter RAM

Offset	Bits	Name	Description	Initialized by
0x0–0x7F	—	Reserved	Internal variables	N/A
0x80–0xBF OR 0x80–0xFF OR 0x80–0x11F	64 or 128 or 160 bytes	Extended Parsing Thread Parameters	Internal variables. Needed if Extended Parsing Mode is enabled (REMODER[EXP]=1). 0x80–0xBF - No external Hash Lookup 0x80–0xFF - External Hash Lookup and the longest LookupKey is 8 bytes 0x80–0x11F - External Hash Lookup and the longest LookupKey is 16bytes Note: These Parameters are need if the UEC is used in Termination mode only. If Interworking packages are used, these parameters need no to be allocated in this area of memory.	N/A

29.5.3.7 Rx Ethernet Mode Register (REMODER)

The Ethernet Mode Register defines extended modes for the UCC Ethernet Port.

offset	0x4															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	EXF	—					VTagOP				VNonTagOP	Res			RQoS	
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	Loss Less FCEn	—	RFSE	EXP	NumOfQueues			—				DXE	DNE	IPCHK	IPAE
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 29-37. UCC Ethernet Mode Register (REMODER)

Table 29-34. REMODER Field Descriptions

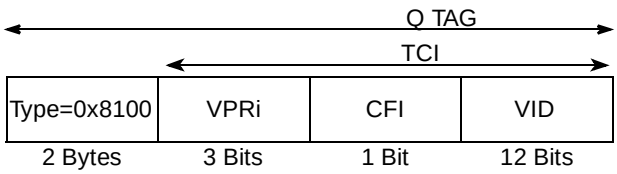
Bits	Name	Description
0	EXF	EXtended Feature Mode 0 Extended features are disabled. 1 Extended features are enabled. These features include: VLAN Tag insertion/removal, IP alignment. Some of the features have separate enable bits. In addition, if EXP=0 (MPC82xx like filtering mode) EXF=0 - One individual address programmed in Section 29.5.1.15, “Station Address Part 1 Register (MACSTNADDR1)” and Section 29.5.1.16, “Station Address Part 2 Register (MACSTNADDR2)” registers. EXF=1 - Five individual addresses as programmed in Section 29.5.1.15, “Station Address Part 1 Register (MACSTNADDR1)” and Section 29.5.1.16, “Station Address Part 2 Register (MACSTNADDR2)” registers, and in Global Rx Parameter RAM AF entry as described in Section 29.5.3.8, “Address Filtering (AF) Field Description.” Refer to Figure 29-6 and Figure 29-8 for relationship between EXP and EXF fields.
1–5	Reserved	Set to zero.
6–9	VTagOP	VLAN Operation for tagged income frames 0000 - No VLAN TAG operation is executed (NOP). 0001 - Replace VID portion of Q Tag. Other fields in TCI are left unchanged. 0010 - If VID=0 replace VID with default value. Other fields in TCI are left unchanged. 0011 - Extract Q Tag from Frame 0100..1111 - Reserved Note: REMODER[EXF] must be set if this field is not equal to zero. Note: Frames shorter than 64 bytes which are received (as programmed in UPSMR[RSH] and MINFLR) do not undergo header manipulation. Note: New value for VID depends on mode of operation. If REMODER[EXP]=0 the new value is taken from GlobalRx Parameter RAM[VID] field (or VID portion of this field). Id REMODER[EXP]=1, the new value of this field (VtagOP) an the value of the VID is taken from the Termination Action Descriptor TCI field (TAD[VTgOP] and TAD[VID]), which is the result of the lookup. 
10	VNonTagOP	VLAN Operation for non-tagged income packets 0 - No VLAN TAG operation is executed (NOP). 1 - Q TAG Insert. A Q Tag is inserted between the SA and the incoming T/L field. The new VID is taken from the default TCI (TCI entry in Global Rx Parameter RAM). REMODER[EXF] must be set if this field is not equal to zero. Note: Frames shorter than 64 bytes which are received (as programmed in UPSMR[RSH] and MINFLR) do not undergo header manipulation.
11–13	Reserved	Set to zero

Table 29-34. REMODER Field Descriptions (continued)

Bits	Name	Description
14–15	RQoS	Receive QoS Mode. valid if REMODER[EXP]=0 00 - Use Default Queue for received Frame from Parameter RAM as programmed in TCI[VPri] field in Global Rx Parameter RAM after L2 translation. See Section 29.5.3.11.1, “Rx Layer 2 QoS Table—L2QT.” 01 - Determine queue number in Receive direction using L2 criteria 10 - Determine queue number in Receive direction using L3 criteria 11 - Reserved See Section 29.4.10, “Quality of Service (QoS)” for details. Note: Frames shorter than 64 bytes which are received (as programmed in UPSMR[RSH] and MINFLR) are placed in queue 0.
16	Res	Set to zero
17	LossLessFCEn	LossLess Flow Control Enable. 0 - LossLess Flow Control feature is disabled 1 - LossLess Flow Control feature is enabled.
18	Reserved	Set to zero
19	RFSE	Receive Firmware Statistics Counters enable (see Section 29.7.4, “Firmware Counters”.) 0 - Do not perform Firmware statistics on received frames. 1 - Perform Firmware statistics on received frames
20	EXP	EXtended Address Parsing Mode 0 Extended Parsing mode disabled (MPC82xx-like filtering mode). Refer to Figure 29-6 for relationship between EXP and EXF fields. 1 Extended Parsing mode enabled. Refer to Figure 29-8 for relationship between EXP and EXF fields.
21–23	NumOfQueues	Number of Rx queues 000 - One Rx queues 001 - Two Rx queue ... 111 - Eight Rx queues
24–26	Reserved	Set to zero
27	Reserved	Set to zero.
28	DXE	Dynamic Maximum Frame length Enable 0 - Disable Dynamic maximum frame length 1 - Enable Dynamic maximum Frame length See Section 29.7.2, “Dynamic Minimum and Maximum Frame Length,” for more details.
29	DNE	Dynamic miNimum Frame length Enable 0 - Disable Dynamic minimum frame length 1 - Enable Dynamic minimum Frame length See Section 29.7.2, “Dynamic Minimum and Maximum Frame Length,” for more details.

Table 29-34. REMODER Field Descriptions (continued)

Bits	Name	Description
30	IPCHK	IP Checksum Check 0 - Do not check IP checksum in IPv4 frames on receive 1 - Check IP checksum in IPv4 frames on receive REMODER[EXF] must be set if this field is not equal to zero.
31	IPAE	IP Address Alignment Enable 0 - Do not align IP address 1 - Shift frame by two bytes from the beginning of the buffer to align the IP address to a four byte boundaries. REMODER[EXF] must be set if this field is not equal to zero.

29.5.3.8 Address Filtering (AF) Field Description

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	IADDR_H															
Offset + 2																
Offset + 4	IADDR_L															
Offset + 6																
Offset + 8	GADDR_H															
Offset + A																
Offset + C	GADDR_L															
Offset + E																
Offset + 10	Reserved															
Offset + 12	TADDR_H															
Offset + 14	TADDR_M															
Offset + 16	TADDR_L															
Offset + 18	Reserved must be zero															
Offset + 1A	PADDR1_H															
Offset + 1C	PADDR1_M															
Offset + 1E	PADDR1_L															
Offset + 20	Reserved must be zero															
Offset + 22	PADDR2_H															
Offset + 24	PADDR2_M															
Offset + 26	PADDR2_L															
Offset + 28	Reserved must be zero															
Offset + 2A	PADDR3_H															
Offset + 2C	PADDR3_M															
Offset + 2E	PADDR3_L															
Offset + 30	Reserved must be zero															
Offset + 32	PADDR4_H															
Offset + 34	PADDR4_M															
Offset + 36	PADDR4_L															
Offset + 38	Reserved															
Offset + 3A	TCI2_Type															
Offset + 3C–3F	Reserved															

Figure 29-38. Address Filtering (AF) Entry Definition

Table 29-35. Address Filtering (AF Field Description)

Offset	Bits	Name	Description
0x0	—	IADDR_H	Individual Address filters high and low used in the MPC82xx compatible filtering mode (REMODER[EXP]=0). Each bit in this entry is recognized by its index in the 64 bit GADDR vector; if set by the 'SET GROUP ADDRESS' command, causes the reception of a frame whose 6 LSBits of the hashed individual MAC address has the bit in the appropriate index set. 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 19.3.1, "QUICC Engine Command Register (CECR)."
0x4	—	IADDR_L	
0x8	—	GADDR_H	Group Address filters high and low used in the MPC82xx compatible filtering mode (REMODER[EXP]=0). Same as IADDR field for MAC addresses with the I/G bit to one.
0xC	—	GADDR_L	
0x10	—	Reserved	Set to zero.
0x12	—	TADDR_H	This entry is programmed by the CPU before a set group address command is issued to the RISC CECR Register in the MPC82xx compatible filtering mode (REMODER[EXP]=0). The QUICC Engine block uses this address to store the appropriate bit in the IADDR or GADDR vectors.
0x14	—	TADDR_M	
0x16	—	TADDR_L	
0x18–0x37	—	PADDR1–4	Additional four Individual Station Address. PADDR_L is the lowest order half-word, and PADDR_H is the highest order half-word. (Valid only if REMODR[EXF]==1). The basic individual station address is programmed in MACSTNADDR registers. Note: If less that four addresses are needed, program in invalid fields the value programmed in MACSTNADDR registers. Example of byte ordering: For ethernet address 12-34-56-78-AB-CD where the LSBit of the first byte (0x12) is transmitted first on the line. where I/G Bit is Bit 15 of PADDR L and U/L bit is Bit 14 of PADDR L PADDR_H = 0xCDAB, PADDR_M = 0x7856, PADDR_L = 0x3412 The byte ordering of the TADDR fields is the same.
0x38	16	Reserved	Set to zero.
0x3A	16	TCI2_Type	The value of the type field in the second QTag in the frame. This value is used in Extended Parsing mode, with the GenerateLookupKey_EthFast PCD.
0x3E–0x3F	—	Reserved	Set to zero.

29.5.3.9 EtherStatsBase Field Description

The EtherStatsBase field defines the pointer of the statistics gathering data structure.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Reserved								EtherStatsBase							
Offset + 2	EtherStatsBase															

Figure 29-39. Static Parameter (SP) Field Entry

Table 29-36. Static Parameter (SP) Field Description

Offset	Bits	Name	Description
0x0	0–7	Reserved	Set to zero
0x0	8–15	EtherStatsBase	**Ethernet Statistics Data Structure Base Address** The Ethernet Statistics counters a placed in the Multiuser RAM at this base address. It is up to the user to allocate enough memory space for all the counters.
0x2	0–15		

29.5.3.10 RxBD Queue data structures

The distributor maintains a queue of perfected RxBDs per every send queue. The queue is maintained in the internal memory. Every thread keep pointer to general BD parameters table that define this queue, see RBDQPTR field in thread PRAM. The following figure and table describe the content of entries within the queue and BD parameters table

Table 29-37. RxBD Parameter Table Description

Offset	Bits	Name	Description	Initialized by
Offset + 0	0–31	**PR0_BDBPTR**	BD ring0 Base pointer. Buffer Descriptor Base Pointer to address in MURAM (internal) for prefetched BDs. The internal BD ring includes 4 prefetched RxBDs.	QE
Offset + 4	0–31	**PR0_BDPTR**	RxBD ring0 pointer. Buffer Descriptor Pointer to next internal BD address.	QE
Offset + 8	0–31	**PR0_EBDBPTR**	External BD ring0 Base pointer. External Buffer Descriptor Base Pointer to address in External Memory (DDR). This is the only user initiated address through the CPU.	CPU
Offset + C	0–31	**PR0_EBDPTR**	External BD ring0 pointer. External Buffer Descriptor Pointer to next BD address in External Memory.	QE
.....	—	—	Rest of BD Rings	CPU/QE
Offset + 70	0–31	**PR7_BDBPTR**	BD ring7 Base pointer. Buffer Descriptor Base Pointer to address in MURAM (internal) for prefetched BDs. The internal BD ring includes 4 prefetched RxBDs.	QE
Offset + 74	0–31	**PR7_BDPTR**	RxBD ring7 pointer. Buffer Descriptor Pointer to next internal BD address	QE
Offset + 78	0–31	**PR7_EBDBPTR**	External BD ring7 Base pointer. External Buffer Descriptor Base Pointer to address in External Memory (DDR). This is the only user initiated address through the CPU.	CPU
Offset + 7C	0–31	**PR7_EBDPTR**	External BD ring7 pointer. External Buffer Descriptor Pointer to next BD address in External Memory.	QE

29.5.3.11 Rx Priority Mapping Tables

The Ethernet controller support up to eight priority queues. The frame priority can be determined by the VLAN priority (See L2QT, Figure 29-40), IP priority (See L3QT, Figure 29-41) or default priority. See

QoS field in Section 29.5.3.7, “Rx Ethernet Mode Register (REMODER).” The incoming frame is placed in the priority queue as programmed in the PRx field of the L2QT or L3QT register.

29.5.3.11.1 Rx Layer 2 QoS Table—L2QT

The Layer 2 QoS Table converts the frame VLAN priority the actual frame priority.

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	PR_0x0				PR_0x1				PR_0x2				PR_0x3			
Offset + 2	PR_0x4				PR_0x5				PR_0x6				PR_0x7			

Figure 29-40. Layer 2 QoS Table - L2QT

1 Offset from L2QT

Table 29-38. L2QT Description

Offset	Bits	Name	Description	Initialized by
Offset + 0	0–3	PR_0x0	If the original VLAN priority is equal to 0x0, it mapped to this field value.	CPU
	4–7	PR_0x1	If the original VLAN priority is equal to 0x1, it mapped to this field value.	CPU
	8–11	PR_0x2	If the original VLAN priority is equal to 0x2, it mapped to this field value.	CPU
	12–15	PR_0x3	If the original VLAN priority is equal to 0x3, it mapped to this field value.	CPU
Offset + 2	0–3	PR_0x4	If the original VLAN priority is equal to 0x4, it mapped to this field value.	CPU
	4–7	PR_0x5	If the original VLAN priority is equal to 0x5, it mapped to this field value.	CPU
	8–11	PR_0x6	If the original VLAN priority is equal to 0x6, it mapped to this field value.	CPU
	12–15	PR_0x7	If the original VLAN priority is equal to 0x7, it mapped to this field value.	CPU

29.5.3.11.2 Layer 3 QoS Table - L3QT

The Layer 3 QoS Table converts the frame IP (TOS/TC) priority the actual frame priority.

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	PR_0x0				PR_0x1				PR_0x2				PR_0x3			
Offset + 2	PR_0x4				PR_0x5				PR_0x6				PR_0x7			
	. . .															
Offset + E	PR_0x38				PR_0x39				PR_0x3A				PR_0x3B			
Offset + 1F	PR_0x3C				PR_0x3D				PR_0x3E				PR_0x3F			

Figure 29-41. Layer 3 QoS Table - L3QT

1 Offset from L3QT

Table 29-39. L3QT Description

Offset	Bits	Name	Description
Offset + 0	0–3	PR_0x0	If the original IP priority is equal to 0, it is mapped to this field value.
	4–7	PR_0x1	If the original IP priority is equal to 1, it is mapped to this field value.
. . .	. . .	. . .	. . .
Offset + F	8–11	PR_0x3E	If the original IP priority is equal to 62, it is mapped to this field value.
	12–15	PR_0x3F	If the original IP priority is equal to 63, it is mapped to this field value.

29.5.3.12 Rx Interrupt Coalescing Table

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Queue0_IntCoalescing															
Offset + 2																
Offset + 4	Queue0_IntCoalescing_cnt															
Offset + 6																
	. . . .															
Offset + 38	Queue7_IntCoalescing															
Offset + 3A																
Offset + 3C	Queue7_IntCoalescing_cnt															
Offset + 3E																

Figure 29-42. Rx Interrupt Coalescing Table

1 Offset from IntCoalescingPTR

Table 29-40. Rx Interrupt Coalescing Table Description

Offset	Bits	Name	Description	Initialized by
Offset + 0	0–31	Queue0_IntCoalescing	Queue0 Interrupt Coalescing max value.	CPU
Offset + 4	0–31	Queue0_IntCoalescing_cnt	Queue0 Interrupt Coalescing counter. User writes: Queue0_IntCoalescing	CPU
. .	—	—	—	—
	—	—	—	—

Table 29-40. Rx Interrupt Coalescing Table Description (continued)

Offset	Bits	Name	Description	Initialized by
Offset + 38	0–31	Queue7_IntCoalescing	Queue7 Interrupt Coalescing max value.	CPU
Offset + 3c	0–31	Queue7_IntCoalescing_cnt	Queue7 Interrupt Coalescing counter. User writes: Queue7_IntCoalescing	CPU

29.5.4 Bus Mode Register (BMR*x*)

This register was named ‘FCR’ in MPC82xx.

Figure 29-43 shows the format of the transmit and receive bus mode registers, which reside at TSTATE[0–7] and RSTATE[0–7].

Bits	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	DTB	BDB

Figure 29-43. Bus Mode Register (BMR*x*)

29.5.5 LossLess Flow Control

BMR*x* fields are described in Figure 29-41.

Table 29-41. BMR*x* Field Descriptions

Bits	Name	Description
0–1	—	Reserved, must be programmed to 0
2	GBL	Global. Indicates whether the memory operation should be snooped. 0 Snooping on the Coherent System Bus (CSB) is disabled. 1 Snooping on the Coherent System Bus (CSB) is enabled.
3–4	BO	Byte ordering. Used to select the byte ordering of the buffer. 00,01,11 Reserved modes. 10 Big-endian byte ordering. As data is sent onto the serial line from the data buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same buffer word. These bits must be set to 10 value.
5	CETM	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See Figure 18-1and Section 5.3.2.6, “Debug Configuration.”

Table 29-41. BMR*x* Field Descriptions (continued)

Bits	Name	Description
6	DTB	Indicates on what bus the data is located. 0 On the coherent system bus (CSB) 1 Reserved. The programming of this bit must be consistent with the System Configuration.See Figure 18-1.
7	BDB	Indicates on what bus the BDs or are located. In Extended Parsing mode see also LookupBMR field in Section 29.6.2.6, “Hash Table Lookup PCDs” and Section 29.8.2, “Add/Remove Entry in Hash Lookup Table.” 0 On the coherent system bus (CSB). 1 Reserved. The programming of this bit must be consistent with the System Configuration. See Figure 18-1.

See section Section 29.4.6.1.1, “Lossless Flow Control,” for description of this feature.

Figure 29-44. LossLess Flow Control Table

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	RxBD0 CPU Next BD Offset															
Offset + 2	RxBD0 CPU Last BD Offset															
Offset + 4	BD0 LossLess FC Threshold															
Offset + 6	Reserved															
.....																
Offset + 38	RxBD7 CPU Next BD Offset															
Offset + 3A	RxBD7 CPU Last BD Offset															
Offset + 3C	BD7 LossLess FC Threshold															
Offset + 3E	Reserved															

BMR*x* fields are described in Figure 29-42.

Table 29-42. LossLess Flow Control Table Field Descriptions

Name	Description
RxBDn CPU Next BD Offset	Offset from the Base Address of the RxBD ring to the next RxBD which will be handled by the CPU
RxBDn CPU Last BD Offset	Offset from the Base Address of the RxBD ring to the last RxBD in the BD Ring
RxBDn LossLess FC Threshold	Minimum number of empty RxBDs in the queue in terms of RxBDs*(Size of RxBD). If the number of empty RxBDs is below this number, the UEC automatically transmits a Flow Control frame.

29.6 Extended Parsing Mode

In this mode the UCC Ethernet controller provides enhancements to the basic filtering mode provided in most Ethernet MACs. The mode is enabled by the user by programming REMODER[EXP]=1. The Global

Parameter RAM 16 byte EXPGlobalParam is programmed to point to the first Parse Command Descriptor (PCD). Additional PCDs follow the first PCD in subsequent addresses.

29.6.1 Extended Parsing Mode Global Parameters

The Extended Parsing Mode Parameters are located at base address programmed in EXPGlobalParam entry in the Rx Global Parameter RAM. See Section 29.6.2, “Parsing Command Descriptor (PCD),” for more details on Extended Parsing mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Reserved								L2PCDPTR							
Offset + 2	L2PCDPTR															
Offset + 4-7	Reserved															
Offset + 8-F	Reserved															

Figure 29-45. Extended Parsing Mode Global Parameters Definition

Table 29-43. Extended Parsing Global Parameters Description

Offset	Bits	Name	Description
0x0	8–15	L2PCDPTR	Pointer to first Parse Command Descriptor base address. The user is responsible for allocation of space in the Multiuser RAM for the PCDs.
0x2	0–15		
0x4-7	—	Reserved	Set to zero.
0x8-F	—	Reserved	Set to zero

29.6.2 Parsing Command Descriptor (PCD)

The Parsing Command Descriptor (PCD) is a data structure programmed by user, which is used for parsing of the frame, generation of LookupKey and for lookup in Lookup Tables. Multiple PCDs may be used on a given frame. The pointer to the first PCD is located in the multiuser RAM at base address L2PCDPTR which is programmed in the Extended Parsing Mode parameter RAM data structure. Each PCD is 8 bytes long. PCD commands are executed sequentially until a match occurs or a last PCD is encountered.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	PCD OPCODE								PCD Specific Parameters							
Offset + 2	PCD Specific Parameters															
Offset + 4																
Offset + 6																

Figure 29-46. PC Opcodes

Figure 29-44 specifies the PCD Opcodes.

Table 29-44. PCD Opcodes

Opcode Name	PCD Opcode	Comments
GenerateLookupKey_ EthFast	0x0	This PCD is used to generate a LookupKey from the L2 frame header fields. Because all other PCD types use a LookupKey this PCD must be the first one to be invoked.
ChangeMask	0x10	This PCD is used to mask portions of the LookupKey. This PCD can be invoked only when a valid LookupKey has been generated by a previous PCD.
StoreLookupKey	0x11	This PCD is used to store the current LookupKey. This PCD is used before the ChangeMask PCD if the original LookupKey is needed at a later stage for some other lookup.
RestoreLookupKey	0x12	This PCD is used to restore the LookupKey before that last masked key. With this PCD it possible to Generate a LookupKey, and then perform lookup on subsets of the key.
FourWayHashLookup	0x20	This PCD is used to perform a lookup in Four Way Hash Lookup mode. This PCD can be invoked only when a valid LookupKey has been generated by a previous PCD.
EightWayHashLookup	0x21	This PCD is used to perform a lookup in Eight Way Hash Lookup mode. This PCD can be invoked only when a valid LookupKey has been generated by a previous PCD.
Last PCD	0x3F	This PCD is the last one. If UPSMR[ECM]=1 receive the frame in queue number 0. In this case MAXD2 bytes of he frame are received.

29.6.2.1 Last PCD

This PCD is used at the end of PCD flow. It specifies the actions taken in case of lookup miss.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	PCD OPCODE=0x3F								Reserved							
Offset + 2	DBM	Reserved						BDM	Reserved							
Offset + 4	Reserved															
Offset + 6																

Figure 29-47. Last PCD Field Descriptions

The PCD fields are described in Table 29-46.

Table 29-45. Last PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+0	0–7	Opcode	0x3F
	8–15	—	Reserved bits. Must be cleared to zero.

Table 29-45. Last PCD Field Descriptions (continued)

Offset	Bits	Name	Description
PCD_BASE+2	0	DBM	Debug Mode 0 - Discard Frame. RMON counter is updated MisMatchDrop. 1 - Receive Frame in queue number 0
	0–5	—	Reserved bits. Must be cleared to zero.
	6	BDM0	RxBD Mark The value of these bits is copied into the RxBD[6].
	7	BDM1	RxBD Mark The value of these bits is copied into the RxBD[7].
	8–15	—	Reserved bits must be cleared to zero.
PCD_BASE+4	—	—	Reserved bits must be cleared to zero.

29.6.2.2 GenerateLookupKey_EthFast PCD

This type of PCD is used to generate a LookupKey from the L2Frame header fields. The LookupKey is used by subsequent PCDs to perform a table Lookup.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCD_BASE + 0	PCD OPCODE=0x00								PCDIDValue							
PCD_BASE + 2	MACdst	MACsrc	TCI1	TCI2	Reserved										SrcPort	PCDID
PCD_BASE+ 4	Reserved															
PCD_BASE + 6	Reserved															

Figure 29-48. Generate L2 LookupKey PCD

The PCD fields are described in Figure 29-46.

Table 29-46. Generate L2 LookupKey PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+0	0–7	OPCODE	OPCODE = 0x0 for Generate Lookup PCD
	8–15	PCDID value	PCDIDvalue User Programmable byte to be optionally used as part of the LookupKey. This field must contain a valid value if PCD[PCDID] bit is set. This field is useful if the same LookupTable is used for many types of LookupKeys or if it necessary to distinguish between a masked LookupKey and a non-masked LookupKey with zeroes in the same bits as the mask.

Table 29-46. Generate L2 LookupKey PCD Field Descriptions (continued)

Offset	Bits	Name	Description
PCD_BASE+2	0	MACdst	Extract 48 bit MAC Destination Address for LookupKey 0 - Disable Extraction of this field 1 - Enable Extraction of this field
	1	MACsrc	Extract 48 bit MAC Source Address for LookupKey 0 - Disable Extraction of this field 1 - Enable Extraction of this field
	2	TCI1	Extract 16 bit TCI1 from Q-Tag 0 - Disable Extraction of this field 1 - Enable Extraction of this field. If a Q-TAG is not present in the frame, a value of zero is used in the LookupKey. Parts of this field may be masked to generate VID or VPri subset, by invoking the PCD with change mask opcode. The type field of the QTag is compared with 0x8100.
	3	TCI2	Extract 16 bit TCI2 from Q-Tag (stacked VLAN tags). 0 - Disable Extraction of this field 1 - Enable Extraction of this field. If a second Q-TAG is not present in the frame, a value of zero is used in the LookupKey. Parts of this field may be masked to generate VID or VPri subset, by invoking the PCD with change mask opcode. The type field of the QTag is compared with the user programmable value programmed in the TCI2_Type field in the AF data structure in the Ethernet Global Rx Parameter RAM.
	4–13	Res	Reserved bits must be cleared to zero.
	14	SrcPort	Add one byte of Serial Number (SNUM) of the source port in LookupKey. This is used to identify the source port (UCC) of the frame. See Section 29.8.2.1, “Explanation on the LookupKey to be Placed in this Host Command” for more details. 0 - Disable 1 - Enable
	15	PCDID	Add user programmable PCDIDValue in LookupKey 0 - Disable 1 - Enable
PCD_BASE+4-7	0–15	—	Reserved bits must be cleared to zero.

The LookupKey is temporarily stored in internal Memory. The header fields are placed in order of parsing; if MACdst field and the VID are enabled for parsing the MACdst field is placed first at bytes 0–5, and the VID is placed starting from byte number 6.

29.6.2.3 Change Mask PCD

This type of PCD is used to mask parts of a LookupKey that was generated by the ‘Generate LookupKey PCD’.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCD_BASE + 0	PCD OPCODE=0x10								Reserved							
PCD_BASE + 2	Reserved												MaskIndex			
PCD_BASE + 4	MaskValue															
PCD_BASE + 6																

Figure 29-49. Change Mask PCD

The PCD fields are described in Figure 29-47.

Table 29-47. Change Mask PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+2	0–11	—	Reserved bits. Must be cleared to zero.
	12–15	MaskIndex	Index in the LookupKey where the four byte mask is applied 000 - Apply Mask at the beginning of the LookupKey on bytes 0,1,2,3 001 - Apply Mask on bytes 4,5,6,7 of the LookupKey 010 - Apply Mask on bytes 8,9,10,11 of the LookupKey 011 - Apply Mask at the end of the LookupKey on bytes 12,13,14,15 Reset Reserved
PCD_BASE+4	—	MaskValue	Bitwise value of Mask that is applied to the LookupKey before the lookup is performed. 0 - Mask appropriate bit of the LookupKey to zero 1 - Do not alter the value of the appropriate bit in the lookupkey
—	—	—	Reserved bits must be cleared to zero.

29.6.2.4 Store LookupKey PCD

This type of PCD is used to store the current the LookupKey. This PCD is useful if the original LookupKey is needed after a ‘ChangeMask’ PCD is applied on the current LookupKey.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCD_BASE + 0	PCD OPCODE=0x11								Reserved							
PCD_BASE + 2	Reserved															
PCD_BASE + 4																
PCD_BASE + 6																

Figure 29-50. Store LookupKey PCD

The PCD fields are described in Figure 29-48.

Table 29-48. Store LookupKey PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+2	0–15	—	Reserved bits. Must be cleared to zero.
PCD_BASE+4	0–15	—	

29.6.2.5 Restore LookupKey PCD

This type of PCD is used to restore the LookupKey that was generated before the ‘Change Mask PCD’.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCD_BASE + 0	PCD OPCODE=0x12								Reserved							
PCD_BASE + 2	Reserved															
PCD_BASE + 4																
PCD_BASE + 6																

Figure 29-51. Restore LookupKey PCD

The PCD fields are described in Figure 29-49.

Table 29-49. Restore LookupKey PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+2	0–15	—	Reserved bits. Must be cleared to zero.
PCD_BASE+4	0–15	—	

29.6.2.6 Hash Table Lookup PCDs

The Hash Table Lookup PCDs are commands used to enable a lookup in a hashed based lookup table that resides either in internal or in external memory. This type of lookup is used for large tables and/or long LookupKeys. The FourWayHashLookup PCD is used when the lookup table has four ways per set, the EightWayHashLookup PCD is used for two four sets tables (in effect eight way hash).

29.6.2.6.1 Four Way Hash Lookup PCD

This type of PCD is used to perform a lookup using the LookupKey that was generated by the ‘Generate LookupKey’ PCD and optionally masked by the ‘Change Mask’ PCD.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>PCD_BASE + 0</td><td colspan="8">PCD OPCODE=0x20</td><td colspan="8">LookupBMR</td></tr>
<tr><td>PCD_BASE + 2</td><td colspan="8">LookupKeySize</td><td colspan="5">Reserved</td><td>EXT</td><td colspan="2">HaskKeySize</td></tr>
<tr><td>PCD_BASE + 4</td><td colspan="15" rowspan="2">LookupTableBase</td></tr>
<tr><td>PCD_BASE + 6</td></tr>
</table>

Figure 29-52. Four Way Hash Lookup PCD

The PCD fields are described in Table 29-46.

Table 29-50. Four Way Hash Lookup PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+0	0–7	OPCODE	Opcode for this PCD
	8–15	LookupBMR	Bus Mode Register for Lookup Table. See Section 29.5.4, “Bus Mode Register (BMRx),” for details. Note that BDB bit determines which bus the LookupTable is located. Note that DTB bit is not used for lookup.

Table 29-50. Four Way Hash Lookup PCD Field Descriptions (continued)

Offset	Bits	Name	Description
PCD_BASE+2	0–7	LookupKeySize	Lookup KeySize This field specifies the size of the LookupKey. The LookupKey parsed by the Generate LookupKey PCD is truncated to the size programmed in this field. 0x3F - 8 bytes 0x5F - 16 bytes Rest - not allowed
	8–10	Reserved	Set to zero
	11	EXT	External Lookup Table 0 - The Lookup Table resides in internal Multiuser RAM 1 - The Lookup Table resides in external Multiuser RAM
	12–15	HashKeySize	HaskKeyMask. This field determines the size of the hash key (such as the number of sets in the Lookup Table). 0000 - 1 bit Hask Key (2 sets in the Lookup Table) 0001 - 2 bit Hask Key (4 sets in the Lookup Table) 0010 - 3 bit Hash Key (8 sets in the Lookup Table) 0011 - 4 bit Hash Key (16 sets in the LookupTable) 0100 - 5 bit Hash Key 0101 - 6 bits Hash Key ..... (all valid values) 1110 - 15 bit Hash Key (32K sets in the Lookup Table) 1111- 16 bit Hash Key (64K sets in the Lookup Table)
PCD_BASE+4	—	LookupTableBase	This field determines the base address of the Lookup Table in internal or external memory as determined by opcode of the PCD. Perform a hash on the LookupKey. Use LookupTableSize field as a pointer to a table located in internal Multiuser RAM. Use HashResult to access the lookup table pointed by LookupTableBase. HashKeySize determines the size of the table (such as the number of bits from the HashResult to be used as an index). A successful match occurs if valid entry is found and exact match to value in table occurs. See Section 29.6.2.6.3, “Four Way and Eight Way Hash Mode Lookup Table Entry (HLUT),” for details on this lookup algorithm.

29.6.2.6.2 Eight Way Hash Lookup PCD

This type of PCD is used to perform a lookup using the LookupKey that was generated by the ‘Generate LookupKey’ PCD and optionally masked by the ‘Change Mask’ PCD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCD_BASE + 0	PCD OPCODE=0x21								LookupBMR							
PCD_BASE + 2	LookupKeySize								Reserved			EXT	HaskKeySize			
PCD_BASE + 4	LookupTableBase															
PCD_BASE + 6																
PCD_BASE + 8	SecondaryLookupTableBase															
PCD_BASE + A																
PCD_BASE + C	Reserved															
PCD_BASE + E																

Figure 29-53. Eight Way Hash Lookup PCD

The PCD fields are described in Table 29-46:

Table 29-51. Eight Way Hash Lookup PCD Field Descriptions

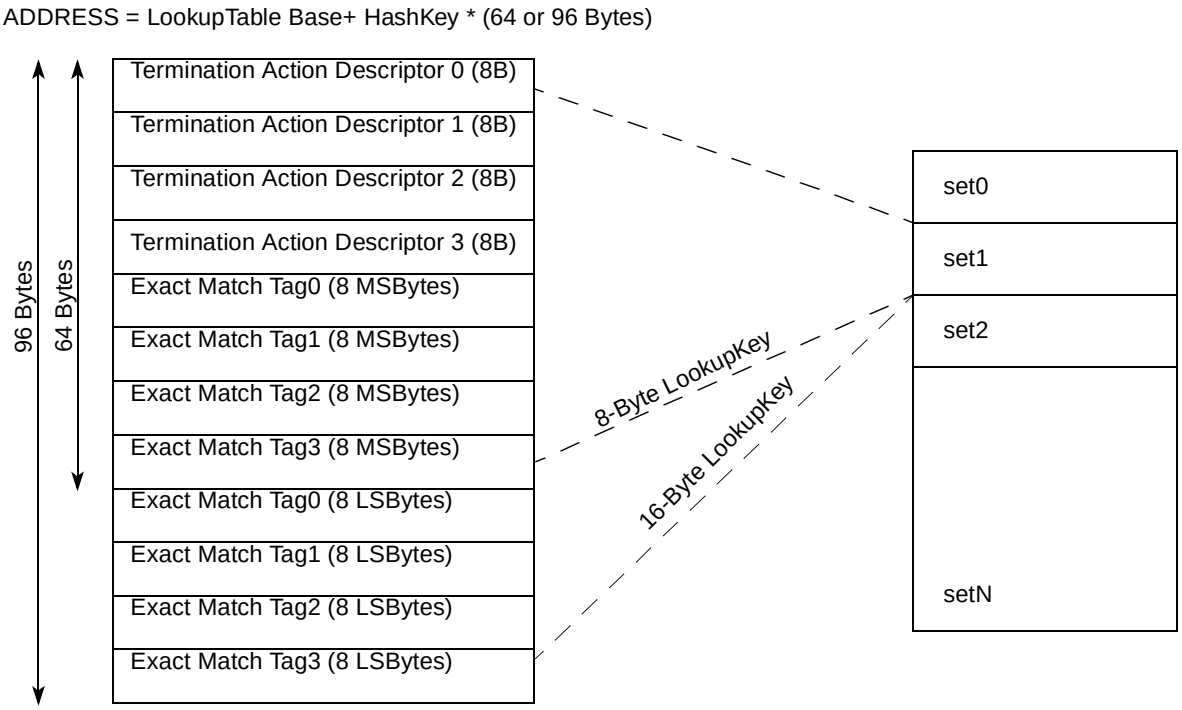
Offset	Bits	Name	Description
PCD_BASE+0	0–7	OPCODE	Opcode for this PCD
	8–15	LookupBMR	Bus Mode Register for Lookup Table. See Section 29.5.4, "Bus Mode Register (BMRx)," for details. Note that BDB bit determines which bus the LookupTable is located. Note that DTB bit is not used for lookup.
PCD_BASE+2	0–7	LookupKeySize	Lookup KeySize This field specifies the size of the LookupKey. The LookupKey parsed by the Generate LookupKey PCD is truncated to the size programmed in this field. 0x3F - 8 bytes 0x5F - 16 bytes Rest - not allowed
	8–10	Reserved	Set to zero
	11	EXT	External Lookup Table 0 - The Lookup Table resides in internal Multiuser RAM 1 - The Lookup Table resides in external Multiuser RAM
	12–15	HashKeySize	HaskKeyMask. This field determines the size of the hash key (such as the number of sets in the Lookup Table). 0000 - 1 bit Hask Key (2 sets in the Lookup Table) 0001 - 2 bit Hask Key (4 sets in the Lookup Table) 0010 - 3 bit Hash Key (8 sets in the Lookup Table) 0011 - 4 bit Hash Key (16 sets in the LookupTable) 0100 - 5 bit Hash Key 0101 - 6 bits Hash Key ..... (all valid values) 1110 - 15 bit Hash Key (32K sets in the Lookup Table) 1111- 16 bit Hash Key (64K sets in the Lookup Table)
PCD_BASE+4	—	LookupTableBase	This field determines the base address of the first four ways Lookup Table in internal or external memory as determined by PCD[EXT] bit. Perform a hash on the LookupKey. Use LookupTableSize field as a pointer to a table located in internal Multiuser RAM. Use HashResult to access the lookup table pointed by LookupTableBase. HashKeySize determines the size of the table (such as the number of bits from the HashResult to be used as an index). A successful match occurs if valid entry is found and exact match to value in table occurs. See Section 29.6.2.6.3, "Four Way and Eight Way Hash Mode Lookup Table Entry (HLUT)," for details on this lookup algorithm.
PCD_BASE+8	—	SecondaryLookupTableBase	This field determines the base address of the Secondary Lookup Table in internal or external memory as determined by PCD[EXT] bit This tale contains additional four ways for the hash lookup. The format of his table is identical to the format of the table pointed by LookupTableBase.
PCD_BASE+C-F	—	Reserved	Set to zero.

29.6.2.6.3 Four Way and Eight Way Hash Mode Lookup Table Entry (HLUT)

The Hash Mode Lookup Table is split into two parts, Termination Action Descriptors and Exact Match Tags.

The table is organized in sets. A set is an entry in the Lookup table, which is indexed by the HashKey. Each set has four ways. Each way contains an Exact Match Tag (8 or 16 bytes) which contains the LookupKey, and a Termination Action Descriptor (TAD) field (8 bytes). The LookupKey is compared to the four ExactMatch Tags in its set corresponding to the four ways, assuming the way is valid (TAD[V]=1). A match yields to table hit. A mismatch in the compare of all four tags, yields to a table lookup miss.

Depending on the application, the user may need either a four way or an eight way hash lookup table. The eight way hash lookup table is composed of two four way lookup tables. In EightwayHash mode lookup, the secondary lookup table is accessed if the LookupKey does not compare to any of the four ExactMatchTags in the first LookupTable. The pointer to this table found in the PCD.



29.6.2.6.4 Exact Match Tags

The exact Match Tag is 8 or 16 bytes long, according to the LookupKeySize. See Section 29.14, “Exact Match Tags Memory Organization” for details.

29.6.2.6.5 Termination Action Descriptor (TAD)

A match event is defined by exact match between the look up key created by the PCD (Or by a PCD pair in the case of masked key) from the income frame to one of the Exact Match Tag fields in the table. The case of non-match is handled by the PCD.

Following a Match event, Frame termination takes place. The termination actions are defined in the Termination Action Descriptor found in the lookup table. Each Exact Match Tag has an Termination Action Descriptor associated with it.

Figure 29-54. Termination Action Descriptor (TAD)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	EXF	V	Rej	Res	Res	Res	VTagOP				VNonVTagOP	Res			RQoS	
Offset + 2	VPriority			CFI	VID											
Offset + 4	—	Reserved														
Offset + 6	Reserved															

Table 29-52. Termination Actions in Hash Mode Lookup Table (HLUT) Entry Field Descriptions

Offset	Bit	Name	Description
0x0–0x1	0	EXF	EXtended Feature Mode 0 Extended features are disabled. 1 Extended features are enabled. These features include: VLAN Tag insertion/removal, IP alignment. Some of the features have separate enable bits. In addition, if EXP=0 (MPC82xx like filtering mode) EXF=0 - One individual address programmed in Section 29.5.1.15, “Station Address Part 1 Register (MACSTNADDR1)” and Section 29.5.1.16, “Station Address Part 2 Register (MACSTNADDR2)” registers. EXF=1 - Five individual addresses as programmed in Section 29.5.1.15, “Station Address Part 1 Register (MACSTNADDR1)” and Section 29.5.1.16, “Station Address Part 2 Register (MACSTNADDR2)” registers, and in Global Rx Parameter RAM AF entry as described in Section 29.5.3.8, “Address Filtering (AF) Field Description.” Refer to Figure 29-6 and Figure 29-8 for relationship between EXP and EXF fields.
	1	V	Valid Entry 0 - The Exact Match Tag in this entry is not valid and should be ignored 1 - The Exact Match Tag in this entry is valid and should be compared to the LookupKey
	2	Rej	Reject Frame 0 - Receive Frame if LookupKey matched the Exact Match Tag 1 - Reject Frame if LookupKey matched the Exact Match Tag
	3	Reserved	Set to zero.

Table 29-52. Termination Actions in Hash Mode Lookup Table (HLUT) Entry Field Descriptions (continued)

Offset	Bit	Name	Description
0x0–0x1	4	Reserved	Set to zero
	5	Reserved	Set to zero
	6–9	VTagOP	0000 - No VLAN TAG operation is executed (NOP). 0001 - Replace VID portion of Q Tag. Other fields in TCI are left unchanged. 0010 - If VID=0 replace VID with default value. Other fields in TCI are left unchanged. 0011 - Extract Q Tag from Frame 0100..1111 - Reserved Note: REMODER[EXF] must be set if this field is not equal to zero. Note: Frames shorter than 64 bytes which are received (as programmed in UPSMR[RSH] and MINFLR) do not undergo header manipulation. Note: The new value of the VID is taken from the Termination Action Descriptor VID field (TAD[VID]), which is the result of the lookup.
	10	VNonTagOP	VLAN Operation for non-tagged income packets 0 - No VLAN TAG operation is executed (NOP). 1 - Q TAG Insert. A Q Tag is inserted between the DA and the income T/L field. The new VID is taken from the default TCI (TCI entry in Global Rx Parameter RAM).
	11–13	Reserved	Set to zero
	14–15	RQoS	RQoS Mode. These bits are used to determine the Rx BD Ring for the CPU. 00 - Use VPri field in TAD[TCI] to determine the queue number, after L2 translation. SeeSection 29.5.3.11.1, “Rx Layer 2 QoS Table—L2QT.” 01 - Determine queue number in Receive direction using L2 criteria 10 - Determine queue number in Receive direction using L3 criteria 11 - Reserved See Section 29.4.10, “Quality of Service (QoS)” for details.
0x2 TCI	0–2	VPri	Vlan Tag priority field. These bits are used to determine the Rx queue if TAD[RQoS]=00.
	3	CFI	Set to zero
	4–15	VID	Vlan Tag
0x4–0x7		Reserved	Set to zero

29.7 Ethernet Statistics

There are three type of statistical counters:

- Hardware Counters
- Firmware Counters
- Software counters.

The Hardware counters are always available to the user. The firmware counters are updated if UPSMR[MON] is set by the user. The Software counters are calculated by the Host CPU, based on the Hardware and Firmware counters. The software counters are specified in this document for completeness.

29.7.1 Features

- Supports IEEE 802.3-2002 Layer management for DTE
  - IEEE 1GE Management Enhancements
  - MAC Entity managed Objects
  - MAC Control entity managed object
  - PAUSE entity managed objects
- Supports IEEE 802.3-2002 Layer Management for Base Band repeaters
- Supports IETF RFC 2819 / STD0059- RMON MIB
- Supports IETF RFC 3635 Managed Object for Ethernet like Interface
- Extensions to the standard statistics is added in order to adapt it to an environment with non-standard length (Jumbo frames, Tagged frames)
- Enhanced transmitter statistics for non-CPU operation

29.7.2 Dynamic Minimum and Maximum Frame Length

The MFLR entry in the Global Parameter RAM defines the length of the largest frame, excluding Q TAG but including FCS, that is still valid. When REMODER[DXE]=1, a tagged frame that has length equals MaxLength+4 considered valid, and a non tagged frame that has length equals MaxLength is the longest that is still considered valid. When REMODER[DXE]=0, any frame longer than MaxLength consider erroneous frame.

For systems with only tagged frames, set REMODER[DXE]=0 and set MaxLength = Max LLC size+4.

The MINFLR entry in the Parameter RAM defines the length of the shortest frame, excluding Q TAG but including FCS, that is still valid frame. The user decides what is considering “Shortest valid frame” in his system: Some users may decide its 68 bytes, some will decide its 64 bytes, and some would decide its dynamic: 68 bytes for tagged frame, 64 bytes for un-tagged frames. In the last case, the user should set the REMODER[DNE] bit and program MinLength to the length of valid un-tagged frame.

See also IEEE 802.3-2002. In 3.2.7 “n” is client data length, and 802.1Q is defined as MAC client and thus Q-Tagg is counted within “n” (such as 64). In figure 3-3 there “Tagged Frame Format” IEEE shows MAC client data excluding Q-Tag (such as 68). In 3.5.7 IEEE adds another comment that leaves interpretation open.

In this last section, a priority-tagged frame considered a tagged frame.

29.7.3 Error Hierarchy

The following error hierarchy is defined by IEEE (In-coherency relative to IETF):

- Frame too Long / Short
- Alignment error
- FCS error
- Length error

29.7.4 Firmware Counters

This section defines the MIB items located in the Multiuser RAM, at the address space defined by SP[EtherStatsBase] field in the Global Parameter RAM. This space in the RAM can be used for any other purpose if Statistics gathering function is disabled. The firmware counters are initialized to zero by the CP

The following error hierarchy is implemented:

Frame too long/short: FrTooLongRx,FrTooShortRx

Alignment error: FrAlignEr

CRC error FrRxF: CSEr

When an error is relevant to more than one counter it is registered only in the highest priority counter.

The user is allowed to reset the Firmware counters dynamically by writing a zero in the corresponding memory location. Because the atomicity of this operation is not guaranteed, the user must read the counter and verify that its value is zero or slightly larger.

29.7.4.1 TX Firmware Counters

Table 29-53. TX Firmware Counters

Address TxEtherStats Base=TESBASE	Name	Width (bit)	Description
TESBASE+0	SiColTx	32	M: Single Collision. Number of Frames that where transmitted OK after a single collision event. Not relevant in full duplex mode.
TESBASE+4	MulColTx	32	M: Multiple Collision. Number of Frames that where involved in more than one collision and than transmitted successfully
TESBASE+8	LateColTxFr	32	Late collision. Counts all frames that were involved in late collision event during frame transmission
TESBASE+C	FrAbortDueCol	32	Frames aborted due to transmit collision. Counts all frames that were aborted either due to repeatedly collision events or due to late collision
TESBASE+10	FrLostInMACTxEr	32	Frames that lost due to internal MAC error transmission, that is not counted on any other counter. Counts all frames that were lost due to any other reason
TESBASE+14	CarrierSenseERTx	32	Carrier Sense Error Counter. Counts the times CS was negated during frame transmission including the case of transmission while CS is negated
TESBASE+18	FrTxOK	32	M: Number of frames transmitted OK
TESBASE+1C	TxFrExcessiveDiffer	32	Conts frames that their differal time was greater than a specified threshold. May
TESBASE+20	TxPkts256	32	Total number of packets (Including bad packets) transmitted that were between 256 (Including FCS length==4) and 511 octets
TESBASE+24	TxPkts512	32	Total number of packets (Including bad packets) transmitted that were between 512 (Including FCS length==4) and 1023 octets
TESBASE+28	TxPkts1024	32	Total number of packets (Including bad packets) Transmitted that were between 1024 (Including FCS length==4) and 1518 octets
TESBASE+2C	TxPktsjumbo	32	Total number of packets (Including bad packets) transmitted that were between 1024 (Including FCS length==4) and MAXLength octets, considering the programmed value MAXLength and the DXE bit defined above. For the case where DXE is set, and MAXLength=1518, all un-tagged packets with length 1518 will be counted by the previous counter. For packets length 1518 or smaller, this counter has the lowest priority.

29.7.4.2 RX Firmware Counters

Table 29-54. RX Firmware Counters

Address EtherStatsBase= RESBASE	Name	Width (bit)	Description
RESBASE+0	FrRxFCSEr	32	M: Number of frames received with CRC error, excluding (does not count) frames which are too long error, too short error, or alignment error, regardless those frames FCS situation.
RESBASE+4	FrAlignEr	32	M:Alignment Errors Counter. Number of frames received that had Alignment errors. Note this counter is not updated for frame shorter that 64 bytes.
RESBASE+8	InRangLenRxER	32	In Rang Length error counter. Counts received frames with L/T field in length mode, that the length of their data field is not equal to the specified length, or that the specified length is less than the minimum LLC frame length regardless of the actual length.
RESBASE+C	OutRangLenRxER	32	Out of Range length Error. Counts all frames with length field greater than the maximum allowed for LLC frame. frames with L/T = T are not included. (Probably reflects a don't care value in the case of a Jumbo frame)
RESBASE+10	FrTooLongRx	32	Counts all received frames with length greater than the programmable parameter MaxLength (see above)
RESBASE+14	Runt	32	Total number of received frames with length smaller than MINLength (as defined above and considering the DNE bit) that incorporates either FCS error or Alignment error, but not frames that would be fine otherwise (Except from the CRC/ALIGN error)
RESBASE+18	VeryLongEventRx	32	Counts all frames received with length greater than MAXLength as defined above and with either FCS error or Alignment error
RESBASE+1C	SymbolErrorRx	32	Number of received frames in which a received error symbol is reported from the PHY layer device during frame reception
REBASE+20	EtherStatsDropRxBsy	32	This includes the total count of drop event due to lack of resources of type BD not ready in the receive process. This counter is a part of the sum composing EtherStatsDropEvent. It can be extracted by summing FrLostInMACTxEr, FrLossInMACRxEr, and EtherStatsDropRxBsy together with the counters below. However, 2819 does not specify exactly which resources it is talking about. (2819 p.17) The UC should count all cases it found BD not ready on Rx. SW should calculate the sum
REBASE+24	Reserved	32	—
REBASE+28	Reserved	32	—
REBASE+2C	MisMatchDrop	32	Counts number of frames dropped due to MAC filtering process, (e.g. Address Mismatch, Type mismatch) and that would otherwise considered good frame that would be transferred to upper layers
REBASE+30	EtherStatsUnderPkts	32	Total number of frames received that were less than 64 octets long but are a good frames otherwise (such as except from the error of being too short, they are good frames)

Table 29-54. RX Firmware Counters (continued)

Address EtherStatsBase= RESBASE	Name	Width (bit)	Description
REBASE+34	EtherStatsPkts256	32	Total number of frames (Including bad frames) received that were between 256 (Including FCS length==4) and 511 octets
REBASE+38	EtherStatsPkts512	32	Total number of frames (Including bad frames) received that were between 512 (Including FCS length==4) and 1023 octets
REBASE+3C	EtherStatsPkts1024	32	Total number of frames (Including bad frames) received that were between 1024 (Including FCS length==4) and 1518 octets
REBASE+40	EtherStatsPktsJumbo	32	Total number of frames (Including bad frames) received that were between 1024 (Including FCS length==4) and MAXLength octets, considering the programmed value MAXLength and the DXE bit defined above. For the case where DXE is set, and MAXLength=1518, all un-tagged frames with length 1518 will be counted by the previous counter. For frames length 1518 or smaller, this counter has the lowest priority.
REBASE+44	FrLossInMACRxEr	32	Frames that lost due to internal MAC error occurred during reception, that is not counted on any other error counter. Counts all frames that were lost due to any error (e.g. OV)
REBASE+48	PausFrRx	32	Counts the total number of PAUSE frames received by this MAC
REBASE+4C	Reserved	—	—
REBASE+50	RxRVLANcnt	32	Counts the total number of frames that their VLAN tag was removed.
REBASE+54	RxRepVLANcnt	32	Counts the total number of frames that their VLAN tag was replaced.
REBASE+58	RxInVLANcnt	32	Counts the total number of frames that VLAN tag was inserted to their header.

29.7.5 UCC Statistics (Hardware Counters)

This section defines the MIB items located at the UCC.

Table 29-55. UCC Statistics

Address	Name	Width (bit)	Description
UCC_Offset+ 0x180	TxframeMiin	32	Total number of frames Transmitted including bad frames that were exactly 64 bytes (Including FCS length==4) Note: Frames with underrun, max collision, late collision errors are not counted
UccOffset+ 0x184	TxPkts65	32	Total number of frames (Including bad frames) transmitted that were between 65 bytes (Including FCS length==4) and 127 octets. Note: Frames with underrun, max collision, late collision errors are not counted

Table 29-55. UCC Statistics (continued)

Address	Name	Width (bit)	Description
UccOffset+ 0x188	TxPkts128	24	Total number of frames (Including bad frames) transmitted that were between 128 (Including FCS length==4) and 255 octets. Note: Frames with underrun, max collision, late collision errors are not counted
UccOffset+ 0x18C	EtherStatsframe64	32	Total number of frames received including bad frames that were exactly in the minimal length (64 bytes)
UccOffset+ 0x190	EtherStatsPkts65	32	Total number of frames (Including bad frames) received that were between 65bytes (Including FCS length==4) and 127 octets
UccOffset+ 0x194	EtherStatsPkts128	24	Total number of frames (Including bad frames) received that were between 128 (Including FCS length==4) and 255 octets
UccOffset+ 0x198	OcTxOK	32	Octet Transmitted OK. Total number of octet residing in frames that where involved successful transmission (30.3.1.1.8)
UccOffset+ 0x19C	PausFrTx	16	Counts the total number of PAUSE control frame transmitted by this MAC (30.3.4.2)
UccOffset+ 0x1A0	MulCastFrTxOK	32	Multicast Frame transmitted OK. Counts all frames that were transmitted successfully with the group address bit set that are not broadcast frames (30.1.1.18)
UccOffset+ 0x1A4	BroadCastFrTxOK	32	Broadcast frames that transmitted OK. Counts all frames transmitted successfully that had DA field equals to the broadcast address (30.3.1.1.19)
UccOffset+ 0x1A8	FrRxOK	32	Number of frames received OK (30.3.1.1.5)
UccOffset+ 0x1AC	OCRxOK	32	Octets received OK. Counts the total number of octets received OK in this probe. (30.1.1.14)
UccOffset+ 0x1B0	EtherStatsOctet	32	Total number of octets received including octets in bad frames. Cannot be calculated by summing the histogram data. Note that it cannot be extracted from the IEEE statistics because the first counts octets only good frames. It has to be implemented in HW because it includes octets in frames that never even reach the UCC
UccOffset+ 0x1B4	MulCastFrRxOK	32	Multicast Frame Received OK. Counts all frames that were received successfully with the group address bit set that are not broadcast frames (3.1.1.21)
UccOffset+ 0x1B8	BroadCastFrRxOK	32	Broadcast frames that received OK. Counts all frames received successfully that had DA field equals to the broadcast address (3.1.1.22)
UccOffset+ 0x1BC–0x1C0	CCR/CMR	32	These registers holds the carry bits and mask of the HW counters. See Section 29.7.7.17, “ Counters Mask Register.”

29.7.6 SW Statistics

The following MIB items can be implemented in SW by the CPU. The QUICC Engine block terminates the control frames. PAUSE control frames are terminated by the MAC, and statistics is gather by the QUICC Engine block. Non-PAUSE control frames are sent to the CPU termination queues.

Table 29-56. MIB

Address	Name	Width (bit)	Description
SW	MACControlFrTx	32	Counts number of control frames either passed to the MAC by higher layers or generated by the MAC (e.g. by host command or RISC command) (30.3.3.3)
SW	MACControlFrRx	32	Counts number of control frames received by the MAC (30.3.3.4)
SW	UnSprtOpcdRx	32	Counts all received control frames (T field = CONTROL) with op-code field that unsupported by this device, including non-valid op-code field, and that are transferred to upper layers as a result (
UC (+SW)	EtherStatsDropEvents	32	This includes the total count of drop event due to lack of resources. The definition is very soft and it can be extracted by summing EtherStatsDropRxBsy, FrLostInMACTxEr, FrLossInMACRxEr,PolicingDrop, QueueManDrop: However, 2819 does not specify exactly which resources it is talking about. (2819 p.17)
SW	EtherStatsPkts	32	Total number of frames including bad frames received. This can be calculated from the histogram data by the CPU
SW	EtherStatsBrdCstPkts	32	Total number of frames received that were directed to a BCST address. Can be calculated by the IEEE item BroadCastFrRxOK
SW	EtherStatsMCSTPkts	32	Total number of frames received that were directed to a MCST address. Can be calculated by the IEEE item MulCastFrRxOK
SW	EnetStatsAlignCRCEr	32	The total number of frames received that does not have length error and have either CRC or alignment error [Can be calculated from IEEE items FrRxFCSEr, FrAlignEr]
SW	EtherStatsOverPkts	32	Total number of frames received that were longer than 1518 octets long [IEEE item FrTooLongRx]
SW	EtherStatsFrgmnt	32	Total number of frames received that were less than 64 octets long and has either CRC or ALGN error [IEEE item RUNt]
SW	EtherStatsJabbers	32	Total number of frames received that were longer than MAXLength (Considering the DXE value and MAXLength bvalue) and had either a bad FCS or Alignment error [can be extracted from IEEE VeryLongEventRx]
SW	EtherStatsCollisions	32	Best Estimation of collision on this ethernet port [IEEE SiColTx, MulColTx, CollisionRx]
SW	LastSRCADDRx	48	Header of BD ring A latch that holds the source address of the last frame received successfully (30.4.3.1.19)
SW	EtherStatsPkts	40	The total number of frames received, including bad frame. Can be calculated by summing all receive distribution counters

29.7.7 Detailed Description of HW Statistics Counters

29.7.7.1 Total Transmit Minimal Length Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TTPML[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TTPML[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-55. Total Transmit Frame Minimal Length (TTPML)

Table 29-57. TTPML Field Descriptions

Bits	Name	Description
0–31	TTPML	Total number of frames Transmitted including bad frames that were exactly in the minimal length (64 for un tagged, 68 for tagged, or with length exactly equal to the parameter MINLength as defined above)

29.7.7.2 Total Transmit Frame 65 to 127 Byte Packet Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TXP65[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TXP65[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-56. Total Transmit Frame 65 to 127 Byte Packet Counter (TXP65)

Table 29-58. TXP65 Field Descriptions

Bits	Name	Description
0–31	TXP65	Total number of frames (Including bad frames) transmitted that were between MINLength (Including FCS length==4) and 127 octets

29.7.7.3 Total Transmit Frame 128 to 255 Byte Packet Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved								TXP128[0–7]							
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TXP128[8–23]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-57. Total Transmit Frame 128 to 255 Byte Packet Counter(TXP128)

Table 29-59. TXP128 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–31	TXP128	Total number of frames (Including bad frames) transmitted that were between 128 (Including FCS length==4) and 255 octets

29.7.7.4 Total Receive Minimal Length Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TRPML[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TRPML[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-58. Total Receive Frame Minimal Length (TRPML)

Table 29-60. TRPML Field Descriptions

Bits	Name	Description
0–31	TRPML	Total number of frames received including bad frames that were exactly in the minimal length (64 bytes)

29.7.7.5 Total Receive Frame 65 to 127 Byte Packet Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RXP65[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	RXP65[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-59. Total Receive Frame 65 to 127 Byte Packet Counter(RXP65)

Table 29-61. RXP65 Field Descriptions

Bits	Name	Description
0–31	RXP65	Total number of frames (Including bad frames) received that were between MINLength (Including FCS length==4) and 127 octets

29.7.7.6 Total Receive Frame 128 to 255 Byte Packet Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved								RXP128[0–7]							
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	RXP128[8–23]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-60. Total Receive Frame 128 to 255 Byte Packet Counter (RXP128)

Table 29-62. RXP128 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–31	RXP128	Total number of frames (Including bad frames) received that were between 128 (Including FCS length==4) and 255 octets

29.7.7.7 Transmit Octet OK Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OcTxOK[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OcTxOK[15–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-61. Transmit Octet OK Counter (OcTxOK)

Table 29-63. OcTxOK Field Descriptions

Bits	Name	Description
0–31	OcTxOK	Octet Transmitted OK. Total number of octet residing in frames that where involved successful transmission

29.7.7.8 Transmit Pause Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TXPF[0–15]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-62. Transmit Pause Frame Counter (TXPF)

Table 29-64. TXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TXPF	Counts the total number of PAUSE control frame transmitted by this MAC

29.7.7.9 Transmit Multicast Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMCA[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TMCA[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-63. Transmit Multicast Frame Counter (TMCA)

Table 0-1TMCA Field Descriptions

Bits	Name	Description
0–31	TMCA	Multicast Frame transmitted OK. Counts all frames that were transmitted successfully with the group address bit set that are not broadcast frames

29.7.7.10 Transmit Broadcast Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TBCA[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TBCA[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-64. Transmit Broadcast Frame Counter (TBCA)

Table 29-65. TBCA Field Descriptions

Bits	Name	Description
0–31	TBCA	Broadcast frames that transmitted OK. Counts all frames transmitted successfully that had DA field equals to the broadcast address

29.7.7.11 Frame Receive OK Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FrRxOK[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	FrRxOK[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-65. Frame Receive OK Counter (FrRxOK)

Table 29-66. FrRxOK Field Descriptions

Bits	Name	Description
0–31	FrRxOK	Number of frames received OK

29.7.7.12 Octet Receive OK Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OCRxOK[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OCRxOK[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-66. Octet Receive OK Counter (OCRxOK)

Table 29-67. OCRxOK Field Descriptions

Bits	Name	Description
0–31	OCRxOK	Octets received OK. Counts the total number of octets received OK in this probe.

29.7.7.13 Receive Total Number Octets Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RxTOC[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	RxTOC[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-67. Receive Total Number Octets Counter (RxTOC)

Table 29-68. OCRxOK Field Descriptions

Bits	Name	Description
0–31	RxTOC	Total number of octets received including octets in bad frames. Cannot be calculated by summing the histogram data. Note that it cannot be extracted from the IEEE statistics because the first counts octets only good frames. It has to be implemented in HW because it includes octets in frames that never even reach the UC

29.7.7.14 Receive Multicast Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RMCA[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	RMCA[16–31]															
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 29-68. Receive Multicast Frame Counter (RMCA)

Table 29-69. RMCA Field Descriptions

Bits	Name	Description
0–31	RMCA	Multicast Frame Received OK. Counts all frames that were received successfully with the group address bit set that are not broadcast frames

29.7.7.15 Receive Broadcast Frame Counter

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RBCA[0–15]															
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	RBCA[16–31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 29-69. Receive Broadcast Frame Counter (RBCA)

Table 29-70. RBCA Field Descriptions

Bits	Name	Description
0–31	RBCA	Broadcast frames that received OK. Counts all frames received successfully that had DA field equals to the broadcast address

29.7.7.16 Counter Carry Register (CCR)

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	CT64	CT65	CT128	CR64	CR65	CR128	COTOK	CTPF	CTMC	CTBC	CRFOK	CROK	CRTO	CRMC	CRBC	—
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	Reserved															
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 29-70. Counters Carry Register (CCR)

Table 29-71. CCR Field Descriptions

Bits	Name	Description
0	CT64	Carry Register Counter TTPML Carry bit
1	CT65	Carry Register Counter TXP65 Carry bit
2	CT128	Carry Register Counter TXP128 Carry bit
3	CR64	Carry Register Counter TRPML Carry bit

Table 29-71. CCR Field Descriptions (continued)

Bits	Name	Description
4	CR65	Carry Register Counter RXP65 Carry bit
5	CR128	Carry Register Counter RXP128 Carry bit
6	COTOK	Carry Register Counter OcTxOK Carry bit
7	CTPF	Carry Register Counter TXPF Carry bit
8	CTMC	Carry Register Counter TMCA Carry bit
9	CTBC	Carry Register Counter TBCA Carry bit
10	CRFOK	Carry Register Counter FrRxOK Carry bit
11	CROK	Carry Register Counter OCRxOK Carry bit
12	CRTO	Carry Register Counter RxTOC Carry bit
13	CRMC	Carry Register Counter RMCA Carry bit
14	CRBC	Carry Register Counter RBCA Carry bit
15	—	Reserved
16–31	—	Reserved

29.7.7.17 Counters Mask Register

offset																
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MT64	MT65	MT128	MR64	MR65	MR128	MOTOK	MTPF	MTMC	MTBC	MRFOK	MROK	MRTO	MRMC	MRBC	—
R/W	R/W															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	Reserved															
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 29-71. Counters Mask Register (CMR)

Table 29-72. CMR Field Descriptions

Bits	Name	Description
0	MT64	Mask Register Counter TTPML Carry bit Mask
1	MT65	Mask Register Counter TXP65 Carry bit Mask
2	MT128	Mask Register Counter TXP128 Carry bit Mask
3	MR64	Mask Register Counter TRPML Carry bit Mask
4	MR65	Mask Register Counter RXP65 Carry bit Mask

Table 29-72. CMR Field Descriptions (continued)

Bits	Name	Description
5	MR128	Mask Register Counter RXP128 Carry bit Mask
6	MOTOK	Mask Register Counter OcTxOK Carry bit Mask
7	MTPF	Mask Register Counter TXPF Carry bit Mask
8	MTMC	Mask Register Counter TMCA Carry bit Mask
9	MTBC	Mask Register Counter TBCA Carry bit Mask
10	MRFOK	Mask Register Counter FrRxOK Carry bit Mask
11	MROK	Mask Register Counter OCRxOK Carry bit Mask
12	MRTO	Mask Register Counter RxTOC Carry bit Mask
13	MRMC	Mask Register Counter RMCA Carry bit Mask
14	MRBC	Mask Register Counter RBCA Carry bit Mask
15	—	Reserved
16–31	—	Reserved

NOTE

When one of the above mask bits are set to zero, the corresponding interrupt bit is allowed to cause interrupt indication on the carry bit in the event register UCCE[carry]

29.8 Ethernet Command Set

The transmit and receive commands described in the following sections are issued to the CECR. Please refer to Section 19.3.1, “QUICC Engine Command Register (CECR)” for detailed description of the command format.

The commands are described in Table 29-73.

Table 29-73. Transmit Commands

Command	Description
STOP TRANSMIT	When used with the Ethernet controller, this command violates a specific behavior of an Ethernet/IEEE 802.3 station. It should not be used.
GRACEFUL STOP TRANSMIT	This command is used to smoothly stop transmission after the current frame finishes sending or undergoes a collision (immediately if there is no frame being sent). UCCE[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 29-73. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	This command enables transmission of characters on the transmit channel. It is expected by the Ethernet controller after a GRACEFUL STOP TRANSMIT command. The Ethernet controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	This command 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. See Section 19.3.1.1, "QUICC Engine Commands."

Receive commands are described in Table 29-74.

Table 29-74. Receive Commands

Command	Description
INIT RX PARAMETERS	This command 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.
INIT TX AND RX PARAMETERS	This command initializes all the receive and transmit parameters in this serial channel parameter RAM to their reset state and should only be issued when the receiver and transmitter are disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the receive and transmit parameters.
SET GROUP ADDRESS	The SET GROUP ADDRESS command is used to set one of the 64 bits of the four individual/group address hash filter registers (GADDR[1–4] or IADDR[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.
ADD/REMOVE ENTRY IN HASH LOOKUP TABLE	This command is used in Extended Parsing mode (REMODER[EXP]=1) in order to add or remove an entry in the Hash Lookup Table. The SBC field in CECR register must contain the value 0x1E0 as defined in the entry named 'General' in the SBC table.
SET LAST RECEIVE REQUEST THRESHOLD	This commands result in programming the DRREQ Register in the UCC. The value in this register determines the timeout value for interrupt coalescing. After this timeout, an interrupt is issued to the CPU also if the interrupt coalescing counter has not reached its value. The user programs the value in CECDR Register.
START FLOW CONTROL	In full duplex mode this command results in sending a flow control frame with MAC parameter as programmed in UEMPT[PT] field. In half duplex mode the UEC send preambles on the line unless data is available for transmit.
STOP FLOW CONTROL	In full duplex mode this command results in sending a flow control frame with MAC parameter set to zero. In half duplex mode the UEC stops sending preambles on the line.
GRACEFUL STOP RECEIVE	This command is used to smoothly stop reception. Once the CPU issues this command, the UEC receiver stops reception after it has smoothly received all frames which are currently being processed. If the command is issued after the UEC has stopped reception, the UEC sets RxGlobalParameterRAM[RXSTPAck]. Therefore, in order to be sure that all receive activity is complete, the CPU must issue this command multiple times, check the RxGlobalParameterRAM[RXSTPAck] until it is set. Reception begins when the E bit of the next BD is set and the RESTART RECEIVE command is issued.
RESTART RECEIVE	This command is used to restart receiving frames after the GRACEFUL STOP RECEIVE is issued.

29.8.1 Init Tx, Init Rx and InitTx and Rx Parameters Command

The INIT TX PARAMETERS Command is issued in order to initialize the Transmit Hardware, and some internal parameters. This command must be issued after the Tx Global Parameter page has been initialized and before the GUMR[ENT] bit is set.

The INIT RX PARAMETERS Command is issued in order to initialize the Receive Hardware, and some internal parameters. This command must be issued after the Rx Global Parameter page has been initialized and before the GUMR[ENR] bit is set.

The INIT TX AND RX PARAMETERS Command is used to initialize Transmit and Receive Hardware, and some internal parameters. This command must be issued after the Rx and Tx Global Parameter page have been initialized and before the GUMR[ENR,ENT] bit are set.

The commands are issued by writing to the CECR register. The command parameters reside in the Multi-user RAM in the InitEnet data structure pointed by the value programmed in the CECDR Register (see Section 19.3.2, “QUICC Engine Command Data Register (CECDR)”). The data structure is 56 bytes long, and is used by the QUICC Engine block for executing the command. Once the command is completed, this data structure may be overwritten. The CECDR register must be programmed before programming the CECR. See Section 19.3.1, “QUICC Engine Command Register (CECR)” for details.

The SBC (Sub Block Code) used in the CECR is taken from Table 19-4, and it corresponds to the UCC running the Ethernet.

The InitEnet Data structure comprises of the data necessary to initialize the UEC. The user programs the InitEnet structure as described in Table 29-75. The table is divided into the Rx and Tx. The user fills in a certain number of entries depending on the Rx and Tx bit rates. The user needs to program part of the Tx and part of the Rx entries in the table or both, depending on the command given. Blank spaces are left in the Data Structure if less than the maximum number of the entries are required or if only the Tx or only the Rx are initialized. The number of entries that need to be initialized depend on the mode programmed in INIT RX PARAMETERS[RGF] or INIT TX PARAMETERS[TGF] fields. The number of entries in the table for Tx is one more than the number of threads, and for Rx is two more that the number of threads. See Section 29.4.4, “Multithreading” for an explanation.

The InitEnet fields are described in Table 29-75.

Table 29-75. InitEnet Command Parameter

Offset	Bits	Name	Description
CECDR+0	0–7	Res	Internal variable. Intialize to value = 0x06
CECDR+1	0–7	Res	Internal variable. Intialize to value = 0x30
CECDR+2	0–7	Res	Internal variable. Intialize to value = 0xFF
CECDR+3	0–7	Res	Intialize to value = 0x00
CECDR+4	0–15	Res	Internal variable. Intialize to value = 0x400.

Table 29-75. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+6	0–7	Res	Intialize to value = 0x00
	8–15	LargestLookupKeySize	Largest Lookup KeySize. Needed only if REMODER[EXP]=1 and ExternalHashLookup PCD is used. This field specifies the size of the largest LookupKey used for any of the ExternalHashLookup performed in Extended Parsing Mode. 0x00 - ExternalHashLookup PCD is not used. (Allocate 64 bytes in Extended Parsing Thread Parameter RAM) 0x3F - 8 bytes (Allocate 64+64 bytes in Extended Parsing Thread Parameter RAM) 0x5F - 16 bytes (Allocate 64+96 bytes in Extended Parsing Thread Parameter RAM) Rest - not allowed
CECDR+8	0–3	RGF	Rx Gigabit or Fast. This bit is used to determine the maximum nominal Rx bit rate running on the UEC. T UEC e data structures are programmed during initialization according to this bit. The user must allocate memory in the Multiuser RAM accordingly. See Section 29.8, “Ethernet Command Set” for more details. 0000 - Reserved 0001 - Suggested value for maximum Rx nominal bit rate of 10/100bps (1 thread, use Rx: Th0_SNUM and Th1_SNUM) 0010 - 2 thread (use Rx: Th0_SNUM, Th1_SNUM and Th2_SNUM) 0011 - Reserved 0100 - Reserved
	4–7	TGF	Tx Gigabit or Fast. This bit is used to determine the maximum nominal Rx bit rate running on the UEC. The UEC e data structures are programmed during initialization according to this bit. The user must allocate memory in the Multiuser RAM accordingly. See Section 29.8, “Ethernet Command Set” for more details. 0000 - Reserved 0001 - Suggested value for maximum Tx nominal bit rate of 10/100bps (1 thread, use Tx: Th1_SNUM) 0010 - 2 thread (use Tx: Th1_SNUM and Th2_SNUM) 0011 - Reserved 0100 - Reserved
	8–25	Rx Global Parameter RAM Page	Base Address of the Global Receiver Parameter RAM Page. The address is aligned to 64 bytes. This field contains bits [11:25] of the address. Bits [8–10] are always zero. The user needs to allocate 256 bytes for this page. The address must be aligned to the page size.
	26–27	Reserved	Initialize to zero
	28–31	RISC Allocation	0000 - Reserved 0001 - RISC 1 0010 - Reserved 0011 - RISC 1

Table 29-75. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+c	0–7	Rx Th0_SNUM	SNUM of a thread chosen for UEC Receiver (Mandatory). Refer to Table 19-17. One of the SNUMs is allocated for this thread; each SNUM is used once in the system.
	8–25	Reserved	Initialize to zero
	26–27	Reserved	Initialize to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+10	0–7	Rx Th1_SNUM	SNUM of a thread chosen for UEC Receiver (Mandatory). Refer to table 4-4 ‘SNUM Table in section 4.5.15. One of the threads is allocated; each thread is used once in the system.
	8–25	Rx Thread1 Parameter RAM Page	Base Address of the Thread Parameter RAM Page. The address is aligned to 128 bytes. This field contains bits [11–25] of the address.Bits [8–10] are always zero.The user needs to allocate 128 bytes for this page. The address must be aligned to the page size.
	26–27	Reserved	Set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+14	0–7	Rx Th2_SNUM	Program if RGF = 0x0010. Refer to description of Rx Th1_SNUM
	8–25	Rx Thread2 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page.
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+18	0–7	Rx Th3_SNUM	Reserved. Refer to description of Rx Th1_SNUM
	8–25	Rx Thread3 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+1c	0–7	Rx Th4_SNUM	Reserved.Refer to description of Rx Th1_SNUM
	8–25	Rx Thread4 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.

Table 29-75. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+20	0–7	Rx Th5_SNUM	Reserved.Refer to description of Rx Th1_SNUM.
	8–25	Thread5 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+24	0–7	Rx Th6_SNUM	Reserved.Refer to description of Rx Th1_SNUM.
	8–25	Rx Thread6 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+28	0–7	Rx Th7_SNUM	Reserved.Refer to description of Rx Th1_SNUM.
	8–25	Rx Thread7 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+2c	0–7	Rx Th8_SNUM	Reserved.Refer to description of Rx Th1_SNUM.
	8–25	Rx Thread8 Parameter RAM Page	Refer to description of Rx Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+30	0–31	Reserved	Initialize to zero
CECDR+34	0–31	Reserved	Initialize to zero
CECDR+38	0–7	Reserved	Initialize to zero.
	8–25	Tx Global Parameter RAM Page	Base Address of the Global Transmitter Parameter RAM Page.The address is aligned to 64 bytes. This field contains bits [11–25] of the address. of the address. Bits [8–10] are always zero. The user needs to allocate 128 bytes for this page. The address must be aligned to the page size.
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.

Table 29-75. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+3c	0–7	Tx Th1_SNUM	Mandatory: SNUM of Transmitter of the UCC running the UEC. Refer to Table 19-17. One of the threads is allocated; each thread is used once in the system.
	8–25	Thread1 Parameter RAM Page	Base Address of the Thread Parameter RAM Page. The address is aligned to 64 bytes. This field contains bits [11–25] of the address.Bits [8–10] are always zero.The user needs to allocate 64 bytes for this page. The address must be aligned to the page size.
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+40	0–7	Tx Th2_SNUM	Use if TGF = 0x0010. Refer to description of Tx Th1_SNUM.
	8–25	Thread2 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+44	0–7	Tx Th3_SNUM	Reserved.Refer to description of Tx Th1_SNUM
	8–25	Thread3 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+48	0–7	Tx Th4_SNUM	Reserved.Refer to description of Tx Th1_SNUM
	8–25	Thread4 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page.
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+4c	0–7	Tx Th5_SNUM	Reserved.Refer to description of Tx Th1_SNUM
	8–25	Thread5 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.

Table 29-75. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+50	0–7	Tx Th6_SNUM	Reserved.Refer to description of Tx Th1_SNUM
	8–31	Thread6 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+54	0–7	Tx Th7_SNUM	Reserved.Refer to description of Tx Th1_SNUM
	8–31	Thread7 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+58	0–7	Tx Th8_SNUM	Reserved.Refer to description of Tx Th1_SNUM
	8–31	Thread8 Parameter RAM Page	Refer to description of Thread1 Parameter RAM Page
	26–27	Reserved	set to zero
	28–31	RISC Allocation	Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+5c	0–7	Reserved	set to zero

29.8.1.1 Init Rx Parameters Command

The INIT RX PARAMETERS Command is issued in order to initialize the Receive Hardware, and some internal parameters. This command must be issued after the Rx Global Parameter page has been initialized and before the GUMR[ENR] bit is set.

29.8.2 Add/Remove Entry in Hash Lookup Table

This command is used to add or remove an entry in the Hash Lookup Table. The command is issued by writing to the CECR register (see Section 19.3.1, “QUICC Engine Command Register (CECR),” and Section 19.3.1.1, “QUICC Engine Commands,””), and to CECDR registers (see Section 19.3.2, “QUICC Engine Command Data Register (CECDR).”)

The SBC field in CECR register must contain the value 0x1E0 as defined in the entry named ‘General’ in the SBC table, see Table 19-4. The CECDR register contains the base address in multiuser RAM where the data structure in Figure 29-72 resides. See Section 29.6.2.6, “Hash Table Lookup PCDs,” for more details.

NOTE

The base address programmed in the CECDR must be aligned to a 64 byte boundary.

If the command execution is not able to remove and entry in the lookup table and HTFI bit is set, CETER[0] is set. If the command execution is not able to add an entry from the lookup table and HTFI bit is set CETER[1] bit is set. An interrupt is issued to the CPU if the corresponding CETMR bit is set. The user needs to ensure that there is no conflict with the usage of Timer1 and Timer2 respectively. See Section 19.4.4, “RISC Timer Event Register (CETER)/Mask Register (CETMR).”

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	ADDE		EXLT	SLTE	HTFI	Reserved			LookupBMR							
Offset + 2	LookupKeySize								Reserved				HashKeySize			
Offset + 4	Lookup Table Base Address															
Offset + 6																
Offset+8	Secondary Lookup Table Base Address															
Offset+A																
Offset+C	Reserved															
Offset+E	Reserved															
Offset+10-17	TAD															
Offset+18-1F	Reserved															
Offset 20-2F	LookupKey (up to 16 bytes)															
Offset 30-3F	Reserved															
Offset + 0x40-0x9F or 0x40-0x7F	Reserved (size depends on LookupKeySize: 16 or 8 bytes respectively)															

Figure 29-72. Set entry in Hash Lookup Table Command Parameters

Table 29-76. Set entry in Hash Lookup Field Descriptions

Offset	Bits	Name	Description
0	0–1	ADDE	Add Entry 00 - Reserved 01 - Remove Entry from Lookup Table. 10 - Add entry in lookup Table 11 - Remove entry (if it exists) and then add entry to LookupTable Note: The remove and then add may be used to guarantee that he same LookupKey does not appear twice in the lookup table.
	2	EXLT	External Lookup Table 0 - Lookup Table resides in internal Multiuser RAM 1 - Lookup Table resides in External Memory
	3	SLTE	Secondary Lookup Table Enable 0 - No secondary lookup table 1 - Secondary lookup table is used if all four ways of the lookup table are filled.
	4	HTFI	Hash Table Full Interrupt Enable 0 - No interrupt is issued if Hash Table is full 1 - Enable CETER[0] or CETER[1] interrupts. The user needs to ensure that there is no conflict with the usage of Timer1 and Timer2 respectively.
	5–7	Reserved	Set to zero
	8–15	LookupBMR	Bus Mode Register. This register defines the bus where the LookupTable resides. See Section 29.5.4, “Bus Mode Register (BMRx).” The BDB bit determines the bus where the LookupTable is resides. Note that the value of this bit must match the value programmed in the relevant HashLookup PCD. See Section 29.6.2.6, “Hash Table Lookup PCDs.”
2	0–7	LookupKeySize	Lookup KeySize This field specifies the size of the LookupKey. The LookupKey parsed by the Generate LookupKey PCD is truncated to the size programmed in this field. 0x3F - 8 bytes 0x5F - 16 bytes Rest - not allowed
	8–11	Reserved	Set to zero
	12–15	HashKeySize	HaskKeyMask. This field determines the size of the hash key (such as the number of sets in the Lookup Table). 0000 - 1 bit Hask Key (2 sets in the Lookup Table) 0001 - 2 bit Hask Key (4 sets in the Lookup Table) 0010 - 3 bit Hash Key (8 sets in the Lookup Table) 0011 - 4 bit Hash Key (16 sets in the LookupTable) 0100 - 5 bit Hash Key 0101 - 6 bits Hash Key ..... (all valid values) 1110 - 15 bit Hash Key (32K sets in the Lookup Table) 1111- 16 bit Hash Key (64K sets in the Lookup Table)
4	—	LookupTableBase	This field determines the base address of the Lookup Table in internal or external memory as determined by EXLT bit.

Table 29-76. Set entry in Hash Lookup Field Descriptions (continued)

Offset	Bits	Name	Description
8	—	Secondary LookupTableOffset	Base address of the secondary lookup table. The secondary lookup table is accessed if SLTE bit is set and the lookup table based at LookupTableBase is full (for insert new entry), or if the entry was not found in the same table (for remove).
C	—	Reserved	Set to zero.
E	—	Reserved	Set to zero.
10–17	—	TAD	Termination Action Descriptor
18–1F	—	Reserved	Set to zero.
Offset 20–2F	—	LookupKey	This is the Lookup Key used to generate the new entry in the Lookup Table, or the LookupKey that needs to be removed from the table. The LookupKey is aligned to the low addresses. For lookup keys shorter than 16 bytes, the user must fill the low order bytes with zeroes.
Offset 30–3F	—	Reserved	Set to zero.
Offset + 0x40–0x9F or 0x40–0x7F	—	Reserved	Set to zero. This space is reserved for temporary storage. Its size (64 or 96 bytes) depends on the size of the LookupKey (8 or 16 bytes). This space is needed in the HashLookupTable is in external memory. If ADDE=10 (add entry), only 64 bytes are needed.

29.8.2.1 Explanation on the LookupKey to be Placed in this Host Command

The LookupKey is comprised of all header fields corresponding to the bits set in the GenerateLookupKey PCD that is used to perform the Lookup using this Hash Lookup Table. The fields in the LookupKey appear in the same order as their respective enable bits in the GenerateLookupKey_EthFast PCD PCD. The LookupKey is padded with zeroes to the right up to its programmed size (8 or 16 bytes). If the PCD has a bit set, that corresponds to afield which is not found in the frame, a value of ‘zero’ replaces the missing field. For example: If in the PCD TCI1 is set, and the frame has not VLAN Tag. the lookupkey contains 2 bytes of ‘zero’ in place of the missing VLAN tag. Therefore if the user wishes to have a match on frame with or without VLAN Tag, two entries in the lookup table must be inserted (one with the desired VLAN Tag, and one with 2 bytes of zeroes in place of the VLAN Tag in the lookup Key).

Regarding the SourcePort and PCDID fields. If the user chooses to include these fields in the lookup key, the following instructions must be followed:

The source port values are taken from Figure 29-77.

Table 29-77. Source Port Values (SNUMs)

Source Port	SNUM
UCC1	0x01
UCC2	0x11
UCC3	0x21

Table 29-77. Source Port Values (SNUMs) (continued)

Source Port	SNUM
UCC4	0x31
UCC5	0x41

The PCDID field is user programmable. For a ‘hit’ in the hash table lookup, the user needs to make sure that the value of the PCDID in the corresponding GenerateLookupKey PCD matches the value programmed in this host command.

29.9 Controlling PHY Links (Management Interface)

The support for MII Ethernet Management can be done by the SPI or by one UCC which can be selected using CMXGCR[SMI]. Control and status to and from the PHY is provided via the two-wire MII management interface described in 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.

The UEC MII registers provide the ability to perform continuous read cycles (called a scan cycle); although, scan cycles are not explicitly defined in the standard. 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 scan].

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 UEC. While enabled, the length of MII management frames are reduced from 64 to 32 clocks. This effectively doubles the efficiency of the interface.

29.10 External Signal Description

29.10.1 Overview

This section defines the UCC Ethernet MAC to chip pin I/O. The UEC network interface supports multiple options. One is the MII option which requires 18 I/O pins and supports both data and a management interface to the PHY (transceiver) device. The MII option supports both 10 and 100 Mbps Ethernet rates. Other interface options are the RMII which is a reduced pin implementation of the MII interface. Table 29-78 provides a list of the external signal properties. The assignment of the signals to the pins of the device is described in the signals chapter, parallel I/O, see Section 3.4, “Parallel I/O Ports.”

Table 29-78. Signal Properties

Name	Function	I/O	Pull Up
RX_CLK	MII —receive clock RMII—not in use	Input	—
COL	MII—collision RMII—not in use	Input	—
CRS	MII—carrier sense RMII—not in use	Input	—
RX_DV	MII—RX_DV RMII—carrier sense data valid (CRS_DV)	Input	—
RXD[3:0]	MII—receive data bits 3–0, (RXD[3:0]) RMII—receive data 1–0 (RXD [1:0])	Input	—
RX_ER	MII—receive error (RX_ER) RMII— receive error (RX_ER)	Input	—
TX_CLK	MII—TX_CLK (transmit clock) RMII—REF_CLK (reference clock) This signal usually comes from an external oscillator or PHY	Input	Passive
TXD[3:0]	MII - transmit data bits 3:0 RMII—transmit data bits 1:0	Output	Passive
TX_EN	MII—TX_EN (transmit data enable) RMII—TX_EN (transmit data enable)	Output	Passive
TX_ER	MII- transmit error RMII—not in use	Output	Passive
Management signals	Refer to MII management signals. Management signals are common to all PHY interface.	Input/Output	Passive

29.10.2 Detailed Signal Descriptions

Table 29-79 provides the UCC Ethernet controller interface signal descriptions.

Table 29-79. Signal Descriptions

Signal	I/O	Description	
RX_CLK	I	Receive Clock from PHY. MII mode—A continuous clock (2.5, 25MHz)	
		State Meaning	—
		Timing	Provides a timing reference for RX_DV, RXD, and RX_ER.
COL	I	Collision	
		State Meaning	Asserted— asserted by PHY on detection of a collision on the medium. Negated—No collision detect in the medium
		Timing	Assertion—Shall remain asserted while the collision condition persists. Negation—Must negate while no collision condition persists. COL is not required to transition synchronously with respect to either TX_CLK or RX_CLK.

Table 29-79. Signal Descriptions (continued)

Signal	I/O	Description	
CRS	I	Carrier Sense	
		State Meaning	Asserted— Either the transmit or receive medium is nonidle. PHY shall ensure that Negated—Both the transmit and receive media are idle.
		Timing	Assertion/Negation—Remains asserted throughout the duration of a collision condition.
RX_DV	I	Receive Data Valid MII mode—When this signal is asserted, PHY is indicating that valid data is present on the MII interface. RMII mode—Represents carrier sense data valid, shall be asserted by PHY when the receive medium is nonidle.	
		State Meaning	Asserted— PHY is presenting recovered and decoded nibbles/octets on the RXD<3:0> bundle.
		Timing	Assertion—Shall remain asserted continuously from RST recovered byte/nibble of the frame through the final recovered data byte/nibble. Negation—Negated prior to RST RX_CLK that follows the final byte/nibble.
RXD[3:0]	I	Receive Data MII mode—RXD[3:0] represents a nibble of data to be transferred from PHY to MAC when RX_DV is asserted. A completely formed SFD must be passed across When RX_DV is not asserted, RXD has no meaning. RMII—RXD[1:0] represents a di-bit of data to be transferred from PHY to MAC when CRS_DV is asserted.	
		State Meaning	RXD is a bundle of data signals (RXD<3:0>). RXD<3:0> are driven by PHY.
		Timing	That transition synchronously with respect to RX_CLK in MII and TX_CLK in RMII
RX_ER	I	Receive Error MII and RMII mode—When RX_ER and RX_DV are asserted, PHY has detected an error in the current frame.	
		State Meaning	Asserted—RX_ER is asserted for one or more RX_CLK periods to indicate to the reconciliation sublayer that an error (e.g., a coding error, or any error that PHY is capable of detecting, and that may otherwise be undetectable at the MAC sublayer) was detected somewhere in the frame presently being transferred from PHY to the reconciliation sublayer. Note: While RX_DV is de-asserted, RX_ER shall have no effect on the reconciliation sublayer.
		Timing	That transition synchronously with respect to RX_CLK in MII /TX_CLK in RMII
TX_CLK	I	Transmit Clock MII mode—continuous clock (2.5 or 25 MHz) which provides a timing reference for TX_EN, TXD, and TX_ER RMII mode— reference clock to TX_EN, TXD, TX_ER, RXD, RX_ER and RX_DV	
		State Meaning	MII - Continuous clock that provides the timing reference for the transfer of TX_EN, TXD, and TX_ER signals from the reconciliation sublayer to PHY. TX_CLK is sourced by PHY.
		Timing	—

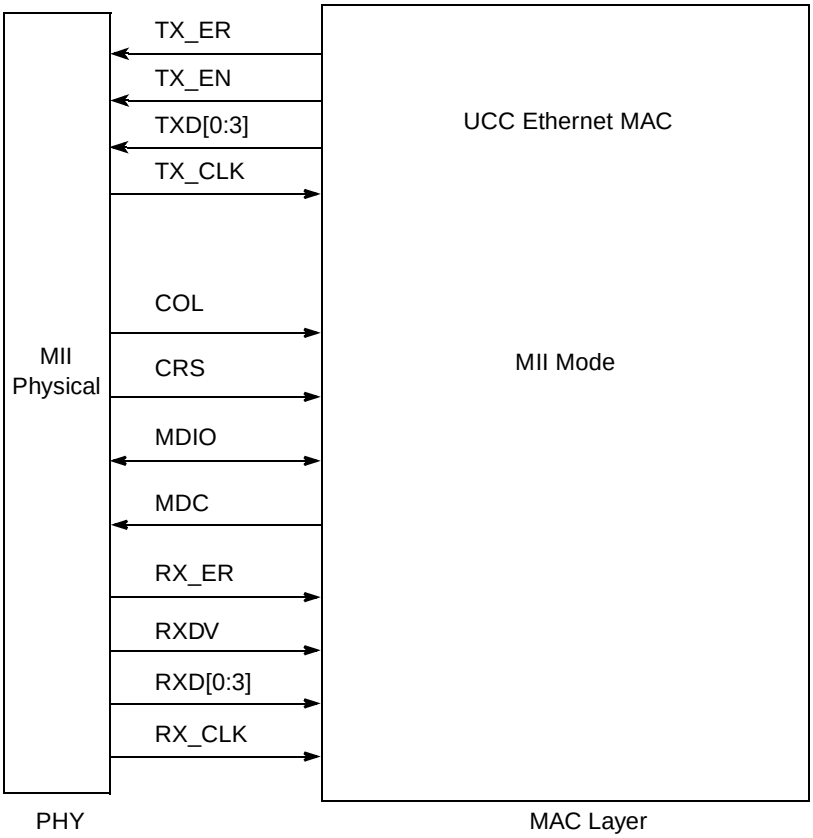
Table 29-79. Signal Descriptions (continued)

Signal	I/O	Description	
TXD[3:0]	O	Transmit Data MII mode—TXD[3:0] represent a nibble of data to be sent from MAC to PHY when TX_EN is asserted and have no meaning when TX_EN is de-asserted. RMII mode—TXD[1:0] represents one complete di-bit of data to be sent from MAC to PHY when TX_EN is asserted and has no meaning when TX_EN is de-asserted.	
		State Meaning	TXD is a bundle of data signals (TXD<0:7>) that are driven by the reconciliation sublayer.
		Timing	TXD<0:7> shall transition synchronously with respect to TX_CLK in MII mode
TX_EN	O	Transmit Enable MII/RMII mode—When this signal is asserted, MAC is indicating that valid data is present on the physical line	
		State Meaning	Asserted— TX_EN indicates that the reconciliation sublayer is presenting nibbles on the MII for transmission. TX_EN shall be negated prior to RST TX_CLK following the nal nibble of a frame. TX_EN is driven by the reconciliation sublayer and shall transition synchronously with respect to TX_CLK.
		Timing	Assertion—synchronously with RST nibble/octet of the preamble and shall remain asserted while all nibbles/octets to be transmitted are presented to the G/MII Negation—negated prior to RST TX_CLK following the nal nibble/octet of a frame.
TX_ER	O	Transmit Error MII mode—Assertion of this signal for one or more clock cycles while TX_EN is asserted shall cause PHY to transmit one or more illegal symbols. Asserting TX_ER has no affect when operating at 10 Mbps or when TX_EN is de-asserted This signal transitions synchronously with respect to TX_CLK.	
		State Meaning	Asserted—PHY shall emit one or more symbols that are not part of the valid data or delimiter set somewhere in the frame being transmitted. The relative position of the error within the frame need not be preserved.
		Timing	TX_ER is asserted for one or more TX_CLK periods while TX_EN is also asserted,
MDC	O	Management data clock In MII/RMII mode, this signal is a clock supplied by the MAC (typically 2.5 MHz). The frequency can be modified by writing to MIICFG[28:31] Note: In order to increase MDC frequency, UPSMR[20] might be used as a prescale by 8	
		State Meaning	MDC is sourced by the station management entity to PHY as the timing reference for transfer of information on the MDIO signal. MDC is an aperiodic signal that has no maximum high or low times.
		Timing	The minimum high and low times for MDC shall be 160 ns each, and the minimum period for MDC shall be 400 ns, regardless of the nominal period of TX_CLK and RX_CLK.
MDIO	I/O	Management data Bi-directional pin to input PHY supplied status during management read cycles and output control during MII management write cycles.	
		State Meaning	MDIO is a bidirectional signal between PHY and STA. It is used to transfer control information and status between PHY and STA.
		Timing	MDIO shall be driven through circuits released to high impedance that enable either STA or PHY to drive the signal.

29.11 UCC Ethernet Controller Connections

29.11.1 Media Independent Interface (MII)

The UCC Ethernet controller implements the IEE 802.3 standard MII interface for Fast Ethernet. Figure 29-73 describes the MII interface signals and shows the basic components, including the signals required to make a UEC module connection with a PHY.



Note: Signal names are according to IEEE 802.3 (total:16 ports).

Figure 29-73. MII MAC-PHY Interface

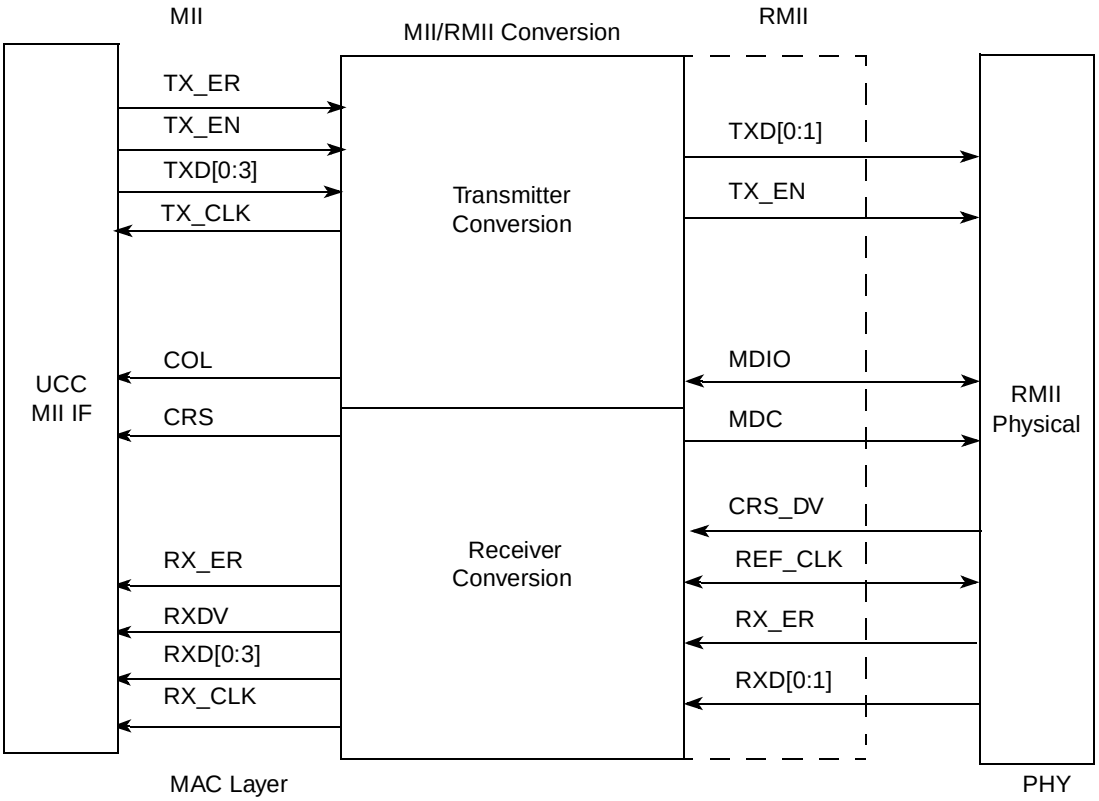
Table 29-80 provides a list of the MII signals.

Table 29-80. MII Signal Descriptions

IEEE Name	I/O	Size	Function	Reference Clock
TX_ER	O	1	Transmit error, asserted by the MAC layer to generate a coding error on the byte currently being transferred over TXD[0:3].	TX_CLK
TX_EN	O	1	Transmit enable, asserted by the MAC sublayer when the first transmit preamble byte is driven over the MII. It remains asserted for the remainder of the frame, up to the last CRC byte.	TX_CLK
TXD[0:3]	O	4	Transmit data	TX_CLK
TX_CLK	I	1	25 MHz clock, which provides the timing reference for the transfer of the TX_EN, TX_ER, and TXD signals.	—
COL	I	1	This signal is asserted on detection of a collision.	TX_CLK
CRS	I	1	This signal is asserted when the transmit or receive medium is not idle.	—
RX_ER	I	1	Receive error, asserted by the PHY layer to indicate an error the MAC can not detect. If asserted during frame reception, indicates a coding error on the frame currently being transferred on RXD[0:3].	RX_CLK
RX_DV	I	1	Receive data valid, asserted by the PHY layer when the first received preamble byte is driving over the MII. It remains asserted for the remainder of the frame, up to the last CRC byte	RX_CLK
RX_CLK	I	1	Receive clock, synchronized all receive signals (RX_DV, RXD, RX_ER)	—
RXD[0:3]	I	4	Receive data	RX_CLK
MDIO	I/O	1	Management data input/output, used to transfer control signals between the PHY layer and the manger entity.	Management Clock
MDC	I	1	Management data clock, the MDIO signal clock reference (25 MHz clock)	—

29.11.2 Reduced Media Independent Interface (RMII)

The UCC Ethernet controller MAC implements the MII/RMII Conversion to achieve pin-out reduce interface. Figure 29-74 shows the basic components of the RMII including the signals required to make a UEC module connection with a PHY.



Note: Signals names are according to 802.3 standard (total: 8 ports).

Figure 29-74. RMII MAC-PHY Interface

Table 29-81 provides a list of the RMII signals.

Table 29-81. RMII Signals

IEEE Name	I/O	Size	Function	Reference Clock	Timing
TX_EN	O	1	Transmit enable, asserted by the MAC sublayer when the first transmit preamble byte is driven over the RMII. It remains asserted for the remainder of the frame, up to the last CRC byte	REF_CLK	max–4 n min–2 n
TXD[0:1]	O	2	Transmit data	REF_CLK	max–4 n min–2 n
REF_CLK	I	1	50-MHz synchronous clock reference for receive, transmit, and control interface.	—	—
RX_ER	I	1	Receive error, asserted by the PHY layer to indicate an error the MAC can not detect. If asserted during frame reception, indicates a coding error on the frame currently being transferred on RXD[1:0].	REF_CLK	max–4 n min–2 n

Table 29-81. RMII Signals (continued)

IEEE Name	I/O	Size	Function	Reference Clock	Timing
CRS_DV	I	1	Receive data valid, asserted by the PHY layer when the first received preamble byte is driving over the RMII. It remains asserted for the remainder of the frame, up to the last CRC byte.	REF_CLK	max–4 n min–2 n
RXD[0:1]	I	2	Receive data	REF_CLK	max–4 n min–2 n
MDIO	I/O	1	Management data input/output, used to transfer control signals between the PHY layer and the manger entity.	Management Clock	—
MDC	I	1	Management data clock, the MDIO signal clock reference (25 MHz clock).	—	—

[1] Signals names are according to 802.3 standard (Total: 4 ports)

29.12 Multiuser RAM usage

The multiuser RAM consumption is dependent on several parameters. This section presents the way the multiuser RAM usage can be calculated according to the setting of these parameters. The calculation is presented by an example, which can be easily modified to reflect the actual parameters values. In addition, each parameter is accompanied by its allowed range, for the user convenience.

Table 29-82 presents the various parameters that impact the multiuser RAM usage.

Table 29-82. Tx and Rx Multi-User RAM Parameters

Parameter	Description	Range	Example for Fast Ethernet (single thread)	Example for Fast Ethernet (two threads)
TxT	Number of Tx threads	1, 2, 4, 6, 8	1	2
TxQ	Number of Tx queues	1 to 8	4	4
RxT	Number of Rx threads	1, 2, 4, 6, 8	1	2
TxVF	Tx Virtual FIFO size [bytes]	408 or more	512	816
RxQ	Number of Rx queues	1 to 8	4	4
RxP	Number of PCDs	0 to 3	0	0
RxL	Rx lookup table size [bytes]	32, 48, 80	0	0
RxVF	Rx Virtual FIFO size [bytes]	408 or more	512	512

Table 29-83 presents the Tx Parameter RAM memory consumption, according to the setting of the appropriate parameters.

Table 29-83. Tx Parameter RAM Usage

Data structure	Size [bytes]	Multiply by	Gigabit Ethernet Example		Fast Ethernet example	
			Value	Total	Value	Total
Global Tx Parameter RAM	128	1	1	128	1	128
Thread Data Structure	136	TxT	4	544	1	136
Queues Data Structure	64	TxQ	4	256	4	256
Scheduler Data Structure	112	1	1	112	1	112
Ethernet Statistics counters	48	1	1	48	1	48
Thread Parameter RAM	64	TxT	4	256	1	64
Total Tx Parameter RAM usage:			—	1344	—	744

Table 29-84 presents the Tx Parameter RAM memory consumption, according to the setting of the appropriate parameters.

Table 29-84. Rx Parameter RAM Usage

Data structure	Size [bytes]	Multiply by	Gigabit Ethernet example		Fast Ethernet example	
			Value	Total	Value	Total
Global Rx Parameter RAM	256	1	1	256	1	256
Thread Data Structure	40	RxT	4	160	1	40
Ethernet Statistics counters	92	1	1	92	1	92
Interrupt Coalescing Counters	8	RxQ	4	32	4	32
Rx BD Queues	16	RxQ	4	64	4	64
Temporary storage for prefetched BDs	32	RxQ	4	128	4	128
Address Filtering Data Structure	8	RxP	2	16	0	0
Lookup table	1	RxL	80	80	0	0
Thread parameter RAM[1]	128	RxT	4	512	1	128
Total Rx Parameter RAM usage:			—	1340	—	740

Note:

1 Size depends on if extended parsing mode is enabled (see Table 29-43,“Extended Parsing Global Parameters Description” and Table 29-33, “Rx Thread Parameter RAM”).

Table 29-85 presents the overall usage of multiuser RAM, by a single Tx port and a single Rx port of Ethernet, according to a specific setting of the various parameters, as presented in Table 29-82.

Table 29-85. Ethernet Multiuser RAM Usage

Data structure	Gigabit Ethernet example size [bytes]	Fast Ethernet example size [bytes]
Tx Parameter RAM usage	1216	744
Tx Virtual FIFO	2048	272
Rx Parameter RAM usage	1340	740
Rx Virtual FIFO	3072	272
Total Multiuser RAM usage:	7676	2028

In order to evaluate the actual multiuser RAM consumption, the user should instantiate the actual values of the various parameters presented in Table 29-82 and follow the example in the tables, Table 29-83 to Table 29-85.

29.13 Header Parsing

The following sections describe in detail the headers being parsed in the Extended Parsing Mode enabled by setting REMODER[EXP]=1. The header extract field in the PCD is programmed to extract one (or more) fields in frame headers as described in the sections below. The shaded fields may be extracted to form a LookupKey.

Note that if the parser is not able to identify one of the frames described in this appendix, the frame is received in Queue number 0, and RxBD[PE] bit is set to mark a parsing error.

29.13.1 Ethernet/802.3 without VLAN

The Next Header PID in an Ethernet or 802.3 frame is located in variable offset from the start of frame, depending on the type of frame. The parser locates the PID for the following types of frames.

29.13.1.1 Ethernet No LLC Header Format

Table 29-86. Ethernet No LLC Header Format

MAC Destination Address	
MAC Destination Address (cont)	MAC Source Address
MAC Source Address (cont)	
EType (>1536)	

29.13.1.2 802.3, 802.2 SAP LLC

This mode is not supported by the parser, and is assumed not to appear in the network. Frames with size > 1536 and NOT SNAP header are forwarded to CPU in queue 0.

Table 29-87. 802.3, 802.2 SAP LLC

MAC Destination Address			
MAC Destination Address (cont)		MAC Source Address	
MAC Source Address (cont)			
Length (>1536)		DSAP	SSAP
Control	L3 header		

29.13.1.3 802.3, 802.2 SNAP LLC

Table 29-88. 802.3, 802.2 SNAP LLC

MAC Destination Address		
MAC Destination Address (cont)		MAC Source Address
MAC Source Address (cont)		
Length (<1536)		DSAP/SSAP = 0xAA-AA
Control = 0x03	OUI 00-00-00	
EType		L3 Header

29.13.2 Ethernet/802.3 with VLAN

29.13.2.1 VTagged Ethernet Encapsulation

Table 29-89. VTagged Ethernet Encapsulation

MAC Destination Address			
MAC Destination Address (cont)	MAC Source Address		
MAC Source Address (cont)			
TPID = 0x8100	PRI	CFI	VLAN ID
EType	L3 Header		

29.13.2.2 VTagged 802.3/802.2 SNAP Encapsulation

Table 29-90. VTagged 802.3/802.2 SNAP Encapsulation

MAC Destination Address			
MAC Destination Address (cont)	MAC Source Address		
MAC Source Address (cont)			
TPID = 0x8100	PRI	CFI	VLAN ID
Length (<1536)	DSAP/SSAP = 0xAA-AA		

Table 29-90. VTagged 802.3/802.2 SNAP Encapsulation

Control = 0x03	OUI 00-00-00
EType	L3 Header

29.13.3 PPPoE+PPP (RFC2516, RFC1661)

Table 29-91. PPPoE+PPP (RFC2516, RFC1661)

PID=88-63 (discovery pkts for CPU) or 88-64 (PPP session)	VER=0x1	TYPE=0x1	CODE=0x0
Session ID (unique with peer MAC address) (PPPSID)	Length of PPPoE payload		
PPP PID (NextHeader Protocol Type)	Payload		

PPPoE+PPP headers are in the Ethernet Payload.

29.13.4 Pv4 (RFC791)

Table 29-92. Pv4 (RFC791)

Version	Header Length	TOS	00	Total Length	
Identification (for fragments)				Flags	Offset
TTL		Protocol Type (PTYPE)		Header Checksum	
IPsrc address					
IPdst address					
Options (optional)					

29.13.5 UDP (RFC768)

Table 29-93. UDP (RFC768)

Portsrc	Portdst
Length	Checksum

29.13.6 TCP (RFC793)

Table 29-94. TCP (RFC793)

Portsrc			Portdst
sequence number			
ack number			
data offset	reserved	flags	window
checksum			urgent pointer
options			padding

29.14 Exact Match Tags Memory Organization

This appendix describes the memory organization of the Exact Match Tags in Hash Mode Table Lookup. See Section 29.6.2.6, “Hash Table Lookup PCDs,” for details on this mode.

The following data structure is used in the Table Lookup set for eight byte LookupKey:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Exact Match Tag 0															
Offset + 2																
Offset + 4																
Offset + 6																
Offset + 8	Exact Match Tag 1															
Offset + A																
Offset + C																
Offset + E																
Offset + 10	Exact Match Tag 2															
Offset + 12																
Offset + 14																
Offset + 16																
Offset + 18	Exact Match Tag 3															
Offset + 1A																
Offset + 1C																
Offset + 1E																

Figure 29-75. Eight Bytes Exact Match Tag (4 Sets)

Table 29-95. Eight Bytes Exact Match Tag Entry Field Descriptions

Offset	Bit	Name	Description
0x0–0x1F	—	ExactMatchTag0-3	Each exact match tag is compared the Lookup Key. If a a match occurs, the Termination Action Descriptor (TAD) is used to determine the action taken on this frame. LookupKey shorter than Eight bytes are aligned to the MSBytes. Unused LSBytes are padded with zeroes.

The following data structure is used in the Table Lookup set for sixteen byte LookupKey:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Exact Match Tag 0 (eight MSBytes)															
Offset + 2																
Offset + 4																
Offset + 6																
Offset + 8	Exact Match Tag 1 (eight MSBytes)															
Offset + A																
Offset + C																
Offset + E																
Offset + 10	Exact Match Tag 2 (eight MSBytes)															
Offset + 12																
Offset + 14																
Offset + 16																
Offset + 18	Exact Match Tag 3 (eight MSBytes)															
Offset + 1A																
Offset + C																
Offset + 1E																
Offset + 20	Exact Match Tag 0 (eight LSBytes)															
Offset + 22																
Offset + 24																
Offset + 26																
Offset + 82	Exact Match Tag 1 (eight LSBytes)															
Offset + 2A																
Offset + 2C																
Offset + 2E																
Offset + 30	Exact Match Tag 2 (eight LSBytes)															
Offset + 32																
Offset + 34																
Offset + 36																
Offset + 38	Exact Match Tag 3 (eight LSBytes)															
Offset + 3A																
Offset + 3C																
Offset + 3E																

Figure 29-76. Sixteen Bytes Exact Match Tag (4 Sets)

Table 29-96. 16 Bytes Exact Match Tag Entry Field Descriptions

Offset	Bit	Name	Description
0x0–0x1F	—	ExactMatchTag0-3 eight MSBytes	Most Significant Bytes: Each exact match tag is compared the Lookup Key. If a a match occurs, the Termination Action Descriptor (TAD) is used to determine the action taken on this frame. LookupKey shorter than sixteen bytes are aligned to the MSBytes. Unused LSBytes are padded with zeroes.
0x20–0x3F	—	ExactMatchTag0-3 eight LSBytes	Least Significant Bytes: Each exact match tag is compared the Lookup Key. If a a match occurs, the Termination Action Descriptor (TAD) is used to determine the action taken on this frame. LookupKey shorter than sixteen bytes are aligned to the MSBytes. Unused LSBytes are padded with zeroes.

29.15 Traffic Shaper Programming Considerations

NOTE

These numerical examples are shown for reference only. They were taken for a Gigabit Ethernet link and can be scaled-down to Fast Ethernet.

For traffic shaper programming considerations, use these definitions:

- AvBR—Average Bit Rate. The average bit rate desired at the port during transmission
- MBL—Maximum Burst Length. The longest wirespeed burst the receiver can absorb. Burst is a group of packets transmitted with minimal IPG (12 Bytes) Between them.
- TSRUnit—The duration of a single TSR count in seconds
- ByteTime—The length it takes to transmit one Byte in data rate equals AvBR in seconds
- TSRByteTime—The length it takes to transmit one Byte in data rate equals AvBR in TSR count units
- MBLInterval = Integer Part of (MBL * TSRByteTime)
- NorTSRByteTime, FracSiz- Normalized TSRByteTime and Fraction Size respectively. Defined by the following two conditions:
  - 1. 256 > NorTSRByteTime > 127
  - 2. NorTSRByteTime = ROUND(TSRByteTime*2^FracSiz)

Example - System requirements are given by the following data rate and burst length:

AvBR = 0.75e9; TSRUnit=1e-6;

MBL = 128 Kbytes

Calculation (Programming model parameters are bolded):

ByteTime = 8/AvBR = 1.06667e-8

TSRByteTime = ByteTime/TSRUnit = 0.0106667

NorTSRByteTime = Round[TSRByteTime*2^14]=ROUND[174.763]=175=0xAF

FracSize=14=0xE

The resulting data rate is:

1/((175/2^14)*1e-6/8)=0.74898Gbps, in the range of 1/256 accuracy.

MBLInterval = ROUND[MBL* TSRByteTime] = ROUND[128KB * 0.0106667] = 1365=0x555

Example 2

AvBR = 64kbps; TSRUnit=1e-6;

MBL = 128KB

Calculation (Programming model parameters are bolded):

ByteTime = 8/AvBR = 0.000125

TSRByteTime = ByteTime/TSRUnit = 125

NorTSRByteTime = Round[TSRByteTime*2^1]=250=0xFA

FracSize=1

The resulting data rate is:

1/((250/2^1)*1e-6/8)=64000.

MBLInterval = ROUND[MBL* TSRByteTime] = ROUND[128KB * 125] = 1.6e007
= 0xF42400

29.15.1 Programming Examples for the Traffic Shaping Characteristics of the Ethernet Tx Distributor

Table 29-97 and Table 29-98 show the user-defined variables.

Table 29-97. Rate Limiting

Name	Type	Units	Description
ExtraBW	8 bits	Set/Clear	Set to indicate a queue's bandwidth will not be taken from the bandwidth allocated
TxASAP	8 bits	Set/Clear	Set to indicate that the queue's frames are never rate limited
NorTSRByteTime	16-bit unsigned integer	Fractions/byte	Time to elapse per byte sent
FracSize	8-bit unsigned integer, 0-32	—	$\log_2$ (fractions per clock)
MBLInterval	32-bit unsigned integer	Clocks	Longest frame burst allowed

Table 29-98. Weighted Fair Queueing

Name	Type	Units	Description
StrictPriorityQ	8 bits	Set/Clear	Set to indicate that a queue will not be subject to weighted fair queuing
WeightFactor	8 8-bit unsigned integers	—	Relative priority of queue (higher value = lower priority)

29.15.1.1 Example: Pass all frames without any Rate Limiting or WFQ

Pass all frames immediately, without either Rate Limiting or Weighted Fair Queuing, prioritized only by queue number.

- Set **StrictPriorityQ** = 0xFF to disable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0xFF to not rate limit any queues
- Set **ExtraBW** = 0xFF to not count bandwidth on any queues
- All other values are ignored

29.15.1.2 Example: Rate Limit all Queues without Maximum Burst Length

Frames are prioritized by queue number, but the total average bandwidth is limited.

Assuming:

Desired average bandwidth: AvBW = 0.75 Gigabits/second

Clock frequency: Freq = 1MHz

Average time elapsed per byte: bytetime = 8 * Freq/AvBW = 0.0106667 cycles/byte

$$\frac{8bits}{byte} \times \frac{1{,}000{,}000cycles}{\sec ond} \times \frac{1\sec ond}{750{,}000{,}000bits} = 0.010\overline{6}\frac{cycles}{byte}$$

The value for NorTSRByteTime should be roughly between 256 and 128 such that:

$$bytetime \times 2^{Fracsize} \approx NorTSRByteTime$$
$$0.010\overline{6} \times 2^{14} \approx 175$$

The value for NorTSRByteTime should be roughly between 256 and 128 such that:

$$bytetime \times 2^{Fracsize} \approx NorTSRByteTime$$
$$0.010\overline{6} \times 2^{14} \approx 175$$

FracSize = 14

NorTSRByteTime = 175

- Set **StrictPriorityQ** = 0xFF to disable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0x00 to rate limit all queues
- Set **ExtraBW** = 0x00 to count bandwidth on all queues
- Set **FracSize** = 0x0E (from above)
- Set **NorTSRByteTime** = 0xAF (from above)
- Set **MBLInterval** - 0xFFFFFFFF to disable burst limiting

29.15.1.3 Example: Rate Limit All Queues with Maximum Burst Length

Frames are prioritized by queue number, but the total average bandwidth is limited.

Assuming:

Desired average bandwidth: AvBW = 0.5 Gigabits/second

Clock frequency: Freq = 10MHz

Maximum burst length: MBL = 128kB

Average time elapsed per byte:

$$bytetime = \frac{8bits}{byte} \times Freq \div AvBW$$

$$\frac{8bits}{byte} \times \frac{10{,}000{,}000cycles}{\sec ond} \times \frac{1\sec ond}{500{,}000{,}000bits} = 0.16\frac{cycles}{byte}$$

The value for NorTSRByteTime should be roughly between 256 and 128 such that:

$$bytetime \times 2^{Fracsize} \approx NorTSRByteTime$$

$$\frac{0.16cycles}{byte} \times \frac{2^{10}\ fractions}{cycle} \approx \frac{164\ fractions}{byte}$$

FracSize = 10
NorTSRByteTime = 164

$$MBLInterval \approx MBL \times bytetime$$

$$131{,}072bytes \times 0.16\frac{cycles}{byte} \approx 20{,}972cycles$$

MBLInterval = 20,972

- Set **StrictPriorityQ** = 0xFF to disable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0x00 to rate limit all queues
- Set **ExtraBW** = 0x00 to count bandwidth on all queues
- Set **FracSize** = 0x0A (from above)
- Set **NorTSRByteTime** = 0xA4 (from above)
- Set **MBLInterval** - 0x000051EC (from above)

29.15.1.4 Example: Disable Rate Limiting Per Queue

Queues 0 and 1 are high priority Voice-over-IP traffic that must always be sent immediately, but the total average bandwidth including all queues is limited.

Assuming:

Desired average bandwidth: AvBW - 0.5 Gigabits/second

Clock frequency: Freq = 10MHz

Maximum burst length: MBL = 128kB

- Set **StrictPriorityQ** = 0xFF to disable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0xC0 to rate limit queues 2–7 and send immediately on queues 0 and 1
- Set **ExtraBW** = 0x00 to count bandwidth on all queues
- Set **FracSize** = 0x0A (from Section 29.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **NorTSRByteTime** = 0xA4 (from Section 29.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **MBLInterval** - 0x000051EC (from Section 29.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)

29.15.1.5 Example: Extra Bandwidth per Queue

Queues 0–5 and 7 belong to a customer that is promised 0.5 Gbps. Queue 6 traffic is for internal ISP use and does not count against the customer’s total bandwidth.

Assuming:

Desired average bandwidth: AvBW - 0.5 Gigabits/second, not including queue 6

Clock frequency: Freq = 10MHz

Maximum burst length: MBL = 128kB

- Set **StrictPriorityQ** = 0xFF to disable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0x02 to rate limit all queues
- Set **ExtraBW** = 0x02 to count bandwidth on all queues except queue 6
- Set **FracSize** = 0x0A (from Section 29.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **NorTSRByteTime** = 0xA4 (from Section 29.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **MBLInterval** - 0x000051EC (from Section 29.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)

29.15.1.6 Example: Weighted Fair Queuing without Rate Limiting

Rate limiting is disabled. Weighted Fair Queuing is enabled on all queues.

Table 29-99. Example Traffic Rate Assumptions

Queue Number	0	1	2	3	4	5	6	7
Traffic Rate	50%	20%	5%	5%	5%	5%	5%	5%

Each WeightFactor should be inversely proportional to the traffic rate desired per queue, with any common factors divided from each value.

$$WeightStatus \propto \frac{1}{trafficrate}$$

Table 29-100. Example Traffic Rate Weight Status

Queue Number	0	1	2	3	4	5	6	7
Traffic Rate	50%	20%	5%	5%	5%	5%	5%	5%
WeightStatus	2	5	20	20	20	20	20	20

- Set **StrictPriorityQ** = 0x00 to enable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0xFF to not rate limit any queues
- Set **ExtraBW** = 0xFF to not count bandwidth on any queues
- Set **WeightStatus** as shown above in Table 29-100

29.15.1.7 Example: Mixed Weighted Fair Queuing and Strict Priority

Rate Limiting is disabled. Queue 4 has the highest priority, then queue 7, then the remaining divided evenly.

Table 29-101. Example Traffic Rate WeightStatus

Queue Number	0	1	2	3	4	5	6	7
Traffic Rate	1/5		1/5	1/5		1/5	1/5	
WeightStatus	1		1	1		1	1	

- Set **StrictPriorityQ** = 0x49 to enable Weighted Fair Queuing on queues 0, 2, 3, 5, and 6
- Set **TxASAP** = 0xFF to not rate limit any queues
- Set **ExtraBW** = 0xFF to not count bandwidth on any queues
- Set **WeightStatus** as shown above in Table 29-101

29.15.1.8 Example: All Concepts

Queues 0–5 belong to a customer who requires 0.8Gbps. Queue 0 is for Voice-over-IP traffic and must be sent at highest priority. Queues 1–5 are weighted as in Table 29-102. Queues 6 and 7 are used internally

by the ISP, and do not count against the customer's usage. The maximum burst for the customer is 1 Megabyte.

Table 29-102. Example Traffic Rate WeightStatus

Queue Number	0	1	2	3	4	5	6	7
Traffic Rate	1/2	1/10	1/10	1/10	1/10	1/0	—	—
WeightStatus	1	5	5	5	5	5	—	—

Assuming:

Desired average bandwidth: AvBW = 0.8 Gigabits/second

Clock frequency: Freq = 1MHz

Maximum burst length: MBL = 128MB

Average time elapsed per byte:

$$bytetime = \frac{8bits}{byte} \times Freq \div AvBW$$

$$\frac{8bits}{byte} \times \frac{1{,}000{,}000cycles}{\sec ond} \times \frac{1\sec ond}{800{,}000{,}000bits} = 0.01\frac{cycles}{byte}$$

The value for NorTSRByteTime should be roughly between 256 and 128 such that:

$$bytetime \times 2^{Fracsize} \approx NorTSRByteTime$$

$$\frac{0.01cycles}{byte} \times \frac{2^{14}\ fractions}{cycle} \approx \frac{164\ fractions}{byte}$$

FracSize = 14
NorTSRByteTime = 164

$$MBLInterval \approx MBL \times bytetime$$

$$134{,}217{,}728bytes \times 0.01\frac{cycles}{byte} \approx 1{,}342{,}177cycles$$

MBLInterval = 1,342,177

- Set **StrictPriorityQ** = 0xFF to disable Weighted Fair Queuing on all queues
- Set **TxASAP** = 0x00 to rate limit all queues
- Set **ExtraBW** = 0x00 to count bandwidth on all queues
- Set **FracSize** = 0x0E (from above equations)
- Set **NorTSRByteTime** = 0xA4 (from above equations)
- Set **MBLInterval** = 0x00147AE1 (from above equations)

Chapter 30
ATM Controller AAL0 and AAL5

30.1 Introduction

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 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), unspecified bit rate (UBR) or UBR+ traffic. To regulate VBR traffic, the APC unit performs a continuous-state leaky bucket algorithm. The APC unit also uses up to eight priority levels to prioritize real-time ATM channels, such as CBR and real-time VBR, over non-real-time ATM channels such as VBR and UBR.

The QUICC Engine block ATM SAR controller applications are as follows:

- ATM line card controllers
- ATM-to-WAN Interworking
- Residential broadband network interface units (NIU) (ATM-to-Ethernet)
- High-performance ATM network interface cards (NIC)
- Bridges and routers with ATM interface

30.1.1 Features

The ATM controller has the following features:

- UTOPIA level II master and slave modes 8 bit
- AAL5, AAL0 protocols
- Up to 253 active VCs internally, and up to 64K VCs using external memory
- TM 4.1 CBR, VBR, UBR, UBR+ traffic types
- VBR type 1 and 2 traffic using leaky buckets (GCRA)
- GBR Guaranteed Bit Rate
- Hierarchical Frame Based scheduling
- Hierarchical Cell Based scheduling
- Unassign cells screening option
- Internal rate transmit mode
- 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 4.1 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
- AAL0 format
  - Receive
    - Whole cell is put in memory
    - Only cell payload is saved in memory
    - CRC10 pass/fail indication
  - Transmit
    - Reads a whole cell from memory
    - CRC10 insertion option
- 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 QUICC Engine block with no host intervention
  - Scalable APC mode
  - APC Flux mode
  - UBR+ prioritized mode
- Receive address look-up mechanism
  - Three modes of address look-up are supported
    - Address compression (including dynamic changes)
    - Mini-CAM look-up
    - Address look-up according to the UDC Extended Address Mode (UEAD)
- 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)
  - Miss inserted cell count

- Memory management
  - RxBD table per VC with option of global free buffer pool for AAL5
  - TxBD table per VC
- UPC
  - Programmable bundling of multiple VCCs to the same UPC, or per connection UPC
  - Up to 1023 Dual leaky buckets
  - Each bucket perform ATM forum's TM4.1 virtual Scheduling GCRA
  - GFR mode as described in ATM forum's TM4.1
  - Detection of cells that violates the traffic agreement.
    - Violating cells can be dropped
    - Violating cells can be tagged
  - Multiple Statistics Counters for cells and frames

30.1.2 ATM Controller—Modes of Operation

The ATM Controller main modes of operations are:

- ATM SAR AAL5, AAL0 protocols
- ATM pace control (APC) unit
  - TM 4.1 CBR, VBR, UBR, UBR+ traffic types
  - Scalable APC mode
  - APC Flux mode
  - UBR+ prioritized mode
  - GCRA Scheduler mode
  - Hierarchical Frame based scheduling
  - Hierarchical Cell Based scheduling
  - GBR -Guaranteed bit rate
- User-defined cells (UDC)
- Performance Monitoring
- Transmit Internal Rate Mode
- Three modes of address look-up are supported
  - Address compression
  - Mini-CAM Address look-up
  - Address look-up according to the UDC Extended Address Mode (UEAD)
- Receive Raw Cell Queue
- OAM Support
  - Virtual Path (F4) Flow
  - Virtual Channel (F5) Flow
- CAS Support

- UPC
- Operation in a 82xx compatible mode and in Multi-Threading mode for high performance/ bandwidth requirements.

30.2 ATM Controller—Functional Description

The following sections provide an overview of the transmitter and receiver portions of the ATM controller. It assumes all the required programming of the Utopia interface and Virtual FIFO are done and memory has been allocated for the Virtual FIFO.

30.2.1 Transmitter Overview

Before the transmitter is enabled, the host must initialize the QUICC Engine block and create the transmit data structure, described in Section 30.3, "ATM Controller—Memory Map." 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.

For DSLAM application in multi PHY systems, in addition to the 82XX APC, which potentially consumes considerable amount of internal memory, especially when the channels rates are spread over a wide range of bandwidth, there is a new traffic shaper which has a smaller memory footprint and which is optimized for such applications. This is a Generic Cell Rate Algorithm (GCRA) scheduler. Number of channels per PHY is limited to 64.

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 UDSR*x*[8–15] when using UTOPIA 8; 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 parameters which resides in the TCT and acts according to the setting in this entry. Each channel could be configured to work in different AAL type, different mode of operation and different scheduling scheme.

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.

When the device in configured to work in "internal rate mode" the Idle cell mode should be disabled, but when using "external rate mode" or when using serial ATM then the Idle cell mode must be enabled.

30.2.1.1 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 30.3.4.2, “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.

30.2.1.2 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.

30.2.2 Receiver Overview

Before the receiver is enabled, the host must initialize the QUICC Engine block and create the receive data structures described in Section 30.3, “ATM Controller—Memory Map.” Data structures varies according to the AAL type and mode of operation. 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 QUICC Engine block from a global free buffer pool. See Section 30.3.9, “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).

30.2.2.1 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.

30.2.2.2 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. There are several options for storing the cell content in the data buffer. In OAM cells the data buffer contains on top of the cell content also time stamp information and the PHY ID on which this OAM cell has been received.

30.2.3 ATM Multi-Threading

The Multi-threading mode of operation is introduced in the QUICC Engine block in order to support higher ATM bit rates. This mode enables parallel processing of ATM traffic thus getting better bus/QUICC Engine block utilization. Operation is kept very similar to the legacy PQ family with some modification to enable parallel processing. This mode is enabled per UCC by configuring the '**ATM_lvl_MT_en**' in the UCC parameter table. (See Table 30-25).

Figure 30-1 illustrates a possible Multi-threading setup. It demonstrates the multi thread structures for UCC Tx or UCC Rx when working in ATM mode. The term Distributor relates to the module which distribute cell handling to the threads. The distributor is directly connected to the serial device (UCC) and its request ID is identical to the UCC Tx or UCC Rx hardware request ID. The threads are processes which performs most of the cell processing work. Each thread has its own ID. Each thread contains information of a single ATM cell belonging to a specific Rx/Tx Channel Code. Operation of the threads is done in parallel. The Terminator is the process which is responsible to keep the correct order of arrival/transmission of the cells to and from the virtual FIFO of the device. Both the receiver and transmitter has its own Terminator SNUM which is allocated statically by the application.

The threads processes are resources available for the QUICC Engine block. These resources could be assigned in a static allocation for each serial device (UCC Rx/UCC Tx) or they could be allocated dynamically to a device upon heavy traffic load conditions. It is recommended that at least one static thread is assigned for each of the UCCs Tx or Rx distributors. Any combination of static/dynamic threads per UCC Tx/Rx is configurable.

Each transmitter/receiver pair requires an internal memory parameters page which in the legacy PQII consists of 0x100 bytes. The page contains very similar information as on the PQ family. The parameters names and programming model functionality is identical to this of the 82XX family. The changes in this new architecture apply to the arrangement of parameters within the page. It has been redefined to enable further expansion and better memory efficiency in future protocols and features. In the minimum configuration it could map to the legacy 82XX model with slight modifications. The distributor process has the traditional 0x100 bytes page and each thread has a 0x80 byte page associated with it. Structure of the page is described in section Section 30.3.1, "Parameter RAM Page Organization."

When the Tx/Rx UCC hardware request is asserted, the distributor module is activated. It performs some activities: scheduling on the transmitter and address look-up on the receiver and identifies the Channel Code of the requestor. Then it triggers operation of a thread for further processing. Threads are allocated dynamically or statically to the distributor. If all threads are busy the procedure in stalled. The objective is to have each thread process a different ATM channel, and by that achieving parallel processing. Identical channel are processed by the same thread. For getting the best performance out of the device multi-threaded operation should be configured only under some conditions:

- If the UPC (Utopia POS Controller) operates in single PHY mode and the number of ATM channels is relatively large and parallelism is achievable.
- In case of using the UPC in multi-PHY mode and a single UCC, it depends on the number of PHYs which are active on the Utopia i/f. Higher number of PHYs increase the efficiency of multi-threading. The user should explore the option of having two or more UCCs handling the same Utopia bus and working in Non multi-threaded operation.

For detailed information about the thread page parameters see Section 30.3.3, “Multi-Threading Structures.”

The minimum amount of processes required for a UCC to work in Multi-threaded mode is the following: Distributor for Rx and Tx (which are hardware-assigned for each UCC), at least one thread for transmitter, at least one thread for receiver, and two threads for terminators which are assigned statically. The user can configure up to 32 threads per one ATM multi thread group, which could serve multiple UCCs. All of these threads use the same ‘common multi thread’ parameters, and can be allocated dynamically according to the system load. See Table 30-20 for the ‘Multi-thread_Common_Base’ parameter. If more than 32 threads are required, the user can allocate additional ATM multi thread group which uses a separate ‘common multi thread’ parameters, in a new UCC page. There is no relationship between these different Multi thread groups, meaning that there will be no dynamic sharing between the threads from one multi thread group to the other.

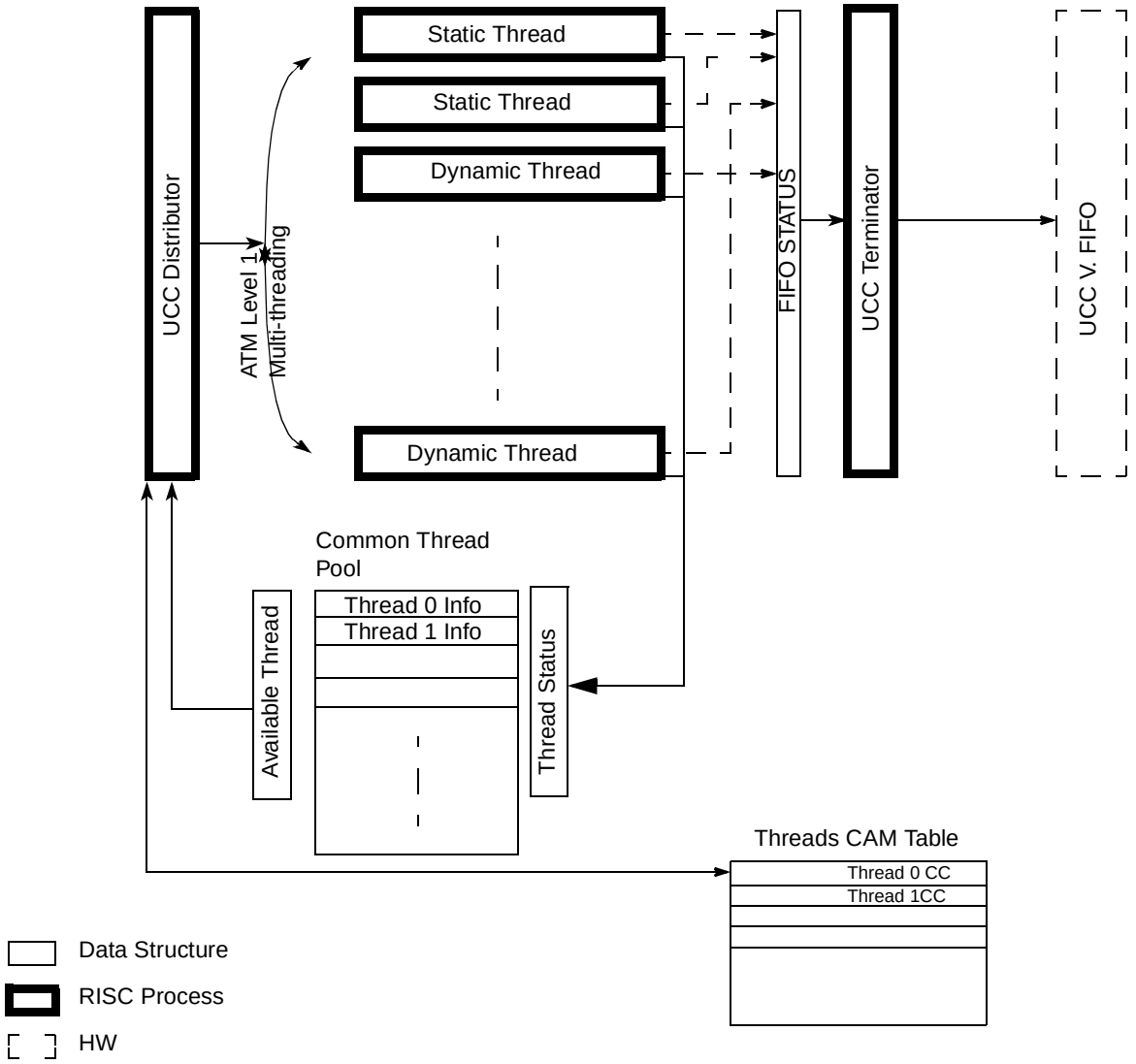


Figure 30-1. ATM UCC Multi-Threading Concept

When the thread is selected, the distributor puts all the relevant cell information in this thread page and triggers the thread.

When the Thread is called, it fetches the cell information from its page, and performs most of the ATM cell processing procedures. Once processing is finished the Terminator is invoked.

The Terminator process is responsible for keeping the sequence of events in the correct order with respect to time and location in the virtual FIFO, since the threads can change the order of the cells.

30.2.4 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 30.2.19, “Performance Monitoring.”

30.2.5 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, UBR) are scheduled at low-priority levels. Up to eight priority levels are available. In GBR mode a channel can resides on two priority levels. One has the parameters of the CBR level rate and one has the UBR level rate.

30.2.5.1 APC Modes and ATM Service Types

The ATM Forum (http://www.atmforum.com) defines the service types described in Table 30-1.

Table 30-1. 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
UBR+	PCR, MCR (peak-and-minimum)	NRT	4
UBR	PCR	NRT	5 (lowest)

For information about cell rate pacing, see Section 30.2.6, “ATM Traffic Type.” For information about prioritization, see Section 30.2.11, “Determining the Priority of an ATM Channel.”

30.2.5.2 APC Unit Scheduling Mechanism

The APC unit consists of an APC data structure in the multi-user RAM for each PHY and a special scheduling algorithm performed by the QUICC Engine block. 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 30.3.6, “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. Up to eight priority levels are available per PHY.

Each scheduling table is divided into time slots, as shown in Figure 30-2. 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.

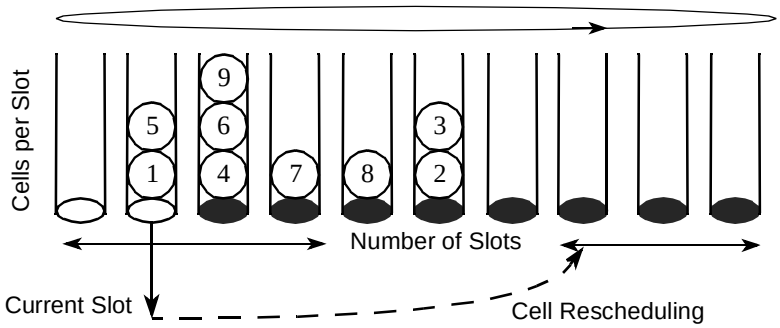


Figure 30-2. APC Scheduling Table Mechanism

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.

Figure 30-3 depicts a schematic view of the ATM scheduling per PHY. Up to eight priority levels are available, each has a the ATM channels assigned which are scheduled according to their contract. Each channels has its parameters stored in a data structure named TCT- Transmit Connection Table and TCTE- Transmit Connection Table Extension.

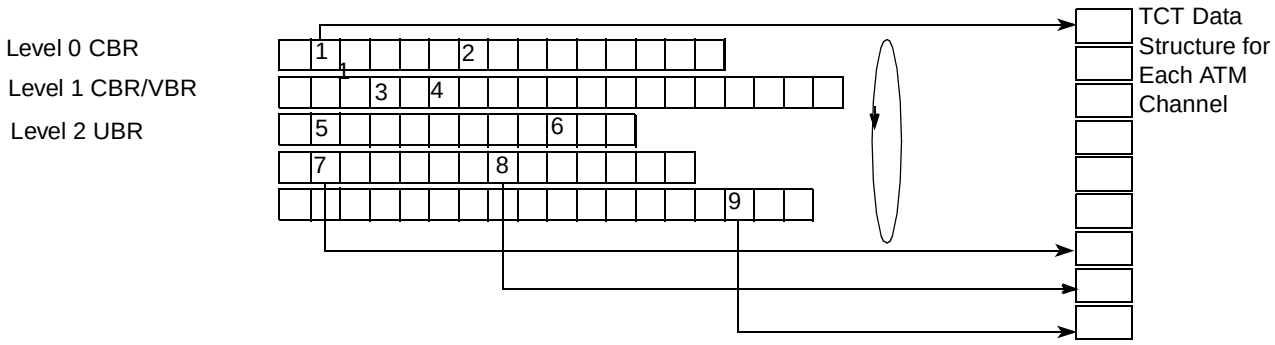


Figure 30-3. ATM Scheduling

30.2.5.3 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.

30.2.5.3.1 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 multi-user RAM increases and more QUICC Engine bandwidth is required.

30.2.5.3.2 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:

$$\text{Max bit rate} = (155.52 \text{ Mbps})/8 = 19.44 \text{ Mbps}$$

$$\text{Min bit rate} = (155.52 \text{ Mbps})/(1024 \times 8) = 18.98 \text{ kbps.}$$

NOTE

In some of the operating modes of the ATM there is an automatic activation of channels into the APC. This operation requires having a minimal size of the APC table which is be 6 slots.

30.2.5.3.3 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.

$$\text{(C)} \qquad \text{Rate [slots]} = \frac{\text{line rate [bps]}}{\text{VC rate [bps]} \times \text{cells per slot}}$$

30.2.6 ATM Traffic Type

The APC uses the cell rate pacing parameters (PCR, SCR, and MCR) to generate CBR, VBR, 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 30.3.4.2, “Transmit Connection Table (TCT).”

30.2.6.1 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.

30.2.6.2 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

30.2.6.3 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 30-4. (The GCRA is also known as the leaky bucket algorithm.)

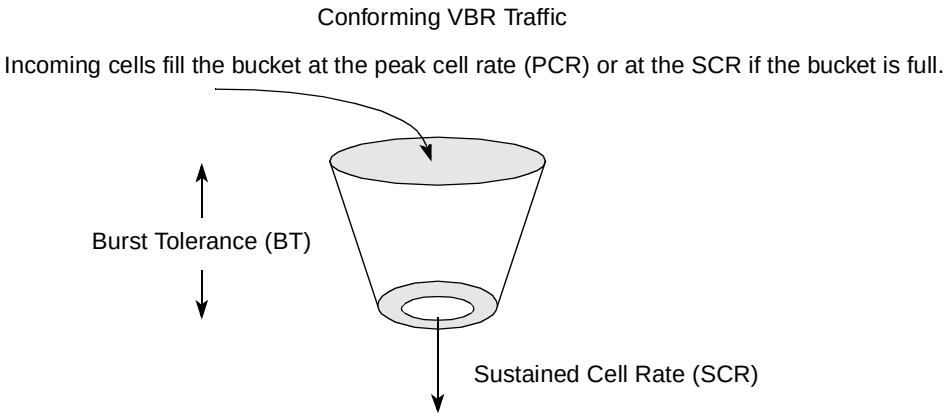


Figure 30-4. VBR Pacing Using the GCRA (Leaky Bucket Algorithm)

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.

30.2.6.3.1 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 30.2.5.3.3, "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.

PCR [slots] = (155.52 Mbps)/(6 Mbps × 8) = 3.24

3.24 = 3 + 0.24 × 256/256 = 3 + 61.44/256 ~ 3 + 62/256

PCR = 3 PCR_FRACTION = 62

SCR [slots] = (155.52 Mbps)/(2 Mbps × 8) = 9.72

9.72 = 9 + (0.72 × 256/256) = 9 + 184.32/256 ~ 9 + 185/256

SCR = 9 SCR_FRACTION = 185

Equation D yields the number of slots the user writes to the channel's TCT[BT].

$$\text{(D)} \qquad \begin{aligned} BT\,[slots] &= (MBS[cells] - 2) \times (SCR[slots] - PCR[slots]) + SCR[slots] \\ &= (1000 - 2) \times ((9+185/256) - (3+62/256)) + (9 + 185/256) \\ &= 6477 \end{aligned}$$

30.2.6.3.2 Handling the Cell Loss Priority (CLP)—VBR Type 1 and 2

The QUICC Engine block 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 30.3.4.2.3, "VBR Protocol-Specific TCTE."

30.2.6.4 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 30.3.4.2.4, "UBR+ Protocol-Specific TCTE***."

30.2.7 Scalable-APC Mode

If an ATM system requires connections with very high variance in bit rates that spans orders of magnitude, the code offers a way to reduce the space required by the APC scheduling table without impacting the scheduling itself. A new APC Mode was defined (Scalable APC) to support Low bit rates connections that would normally require very large APC scheduling tables.

Activating the Scalable APC mode is done by setting the bit SCL_EN in the control slot of the APC, see Table 30-45. The Scalable mode parameters, although transparent to the application, are placed in the TCTE table of the selected ATM channel. If any connection requires a Scalable-APC mode, the TCTE bit in the APC control slot has to be set by the user (if channel number> 255) and so is the SCL_EN bit. It is the application responsibility to clear offsets 0x1C - 0x1F in the TCTE tables of all the connections which resides in the APC scheduling table for proper operation.

30.2.7.1 Scalable-APC Mechanism

In APC scalable mode the user can put in the TCT values of PCR, SCR, MCR and OOBR which are actually bigger than the APC scheduling table size without affecting the scheduling of those channels.

For example: OC-3 PHY (155.52Mbps). ATM 32Kbps CBR connection and CPS=2.

Non APC scalable Mode:

PCR = 155.52Mbps/(32Kbps*CPS) =2430 Slots.

Number of slot in Priority Table = 155Mbps/(32Kbps*CPS) + 1 = 2431 Slots which occupies 4.75 KBytes in DPR.

The value of the CPS is determined by the fastest channel in the system as described in the user's manual. So if the fastest channel doesn't require a small value of CPS it could also be a mean to reduce the APC table size.

APC scalable Mode:

The user may choose an APC table which is smaller from the previous one.

For example: APC table will consist of 300 slots, such as a factor of around 1:8.

The system will behave in the following manner:

As a channel is being scheduled for transmission and the scalable mode is enabled the selected rate for rescheduling is examined and stored in a shadow location. If this value is higher than the NOS-1 (NOS-Number of APC slots) value it indicates we have to "scale" this channel so it is rescheduled to the furthermost point in the APC such as NOS-1 slots away from its current slot. The shadow value is decremented by the NOS-1 value. From that point and on, each time this channel is being selected for transmission the shadow value is examined. As long as its value is greater than the NOS-1 value it will be reschedule with this value, and the cell will not be transmitted. Once the shadow value is smaller than the NOS-1, the rescheduling will be according to the shadow value and on the next time a cell will be transmitted.

The values of the rates on the TCT are programmed as if the APC size is big and the accuracy of the APC is not affected by the change. If for example the SCR value is bigger than the APC size but the PCR is smaller, the scaling would be performed only on the SCR while the PCR will be rescheduled without scaling.

It is important to state that using this mode will have an impact on performance of the QUICC Engine block as channels will be scheduled in the APC without transmitting any data. Bus load will also be increase as there is a need to fetch and update the TCTE parameters even for CBR connections. It also could increase the delay variation when the APC is loaded with a high number of active channels.

30.2.8 APC Flux Compensation

For variable bit rate PHY (for example, radio link), which can fall below the transmission bandwidth defined by the APC internal rate timers, the ATM APC does not use the remaining bandwidth for higher priority traffic only. It "slows down" overall and all ATM channels in all priority levels are affected alike. As a solution to this problem a mechanism called "APC Flux Compensation" was defined. This mechanism is also useful when there is a multi -level APC which one of the lower priority level is oversubscribed with many UBR channels of which many are in idle state meaning there is nothing to transmit in this channels. This situation could cause a delay in servicing the highest priority CBR traffic since checking all those channels is time consuming process which could cause starvation of the highest priority APC level. Using this mechanism causes the CBR high priority APC level to regain the transmission.

When this mode is enabled, by setting Scheduler_MODE[AFC] bit (see Table 30-43) a QUICC Engine Time-Stamp register, CETSR, is set up by programming the CETSCR register, see Section 19.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)."The user should set the value of the expected delta time per APC slot, in the APC_SLOT_DUR_VAL field in the Scheduler Parameter Table, see Table 30-42. This value represents the time needed for transmitting Cell per slot (CPS) ATM Cells on the PHY according to CETSCR pre-scale value.

For example: For a variable bit rate PHY, with a maximum rate set to 50Mbps, with CPS=2 and with CETSCR register programmed to 1 uS. period (For CETSCR which is configured for different period the APC_SLOT_DUR_VAL is different)

APC_SLOT_DUR_VAL= (53 × 8 bit per cell/ Max_Rate) × CPS

=(424b/50Mbps) × 2 = 16.96 us.

The user should program the parameters as follow:

APC_SLOT_DUR_VAL_INT = 0x00000010

APC_SLOT_DUR_VAL_Frac = 0.96 × 2^16 = 0xF5C2.

Whenever the APC Tx flow starts, it captures the CETSR value and compares it to the previous time it got serviced. This value is recorded on each APC service. If the delta time between the current CETSR and the recorded time is bigger than the APC_SLOT_DUR_VAL, then the APC RPTRs of all APC Priorities Tables for this PHY, see Table 30-44, will be advanced by the integer value of: (CETSR - APC_CETSR_LAST_VAL)/(APC_SLOT_DUR_VAL). In case where there is a difference which is a fraction of the APC_SLOT_DUR_VAL it will be recorded by the QUICC Engine block and on each transmit operation the accumulated fraction will be examined and if the value crosses the value of APC_SLOT_DUR_VAL the CETSR value will be advanced by one more entry. In this way there is a continuous compensation for any time drift on the PHY.

In cases of a big timing jitter, advancing the RPTRs would be limited by the value of APC_CETSR_Max_ADV in the Scheduler Parameter Table, see Table 30-43. When the PHY is slowed down for a long period of time and has gained a large credit, such as the RPTR should be advanced by a large number, and the APC scheduling table is not very loaded with channels, advancing the RPTR could be such that it is bigger than some of the channels PCR value. In such a case, this channel could burst in a rate which is higher than the contract rate of this channel when rescheduled by the APC. In order to prevent it from happening, after advancing RPTR to its correct location, the APC Flux mechanism reschedules such a channel to an APC slot which is bigger than the slot pointed by the new RPTR.

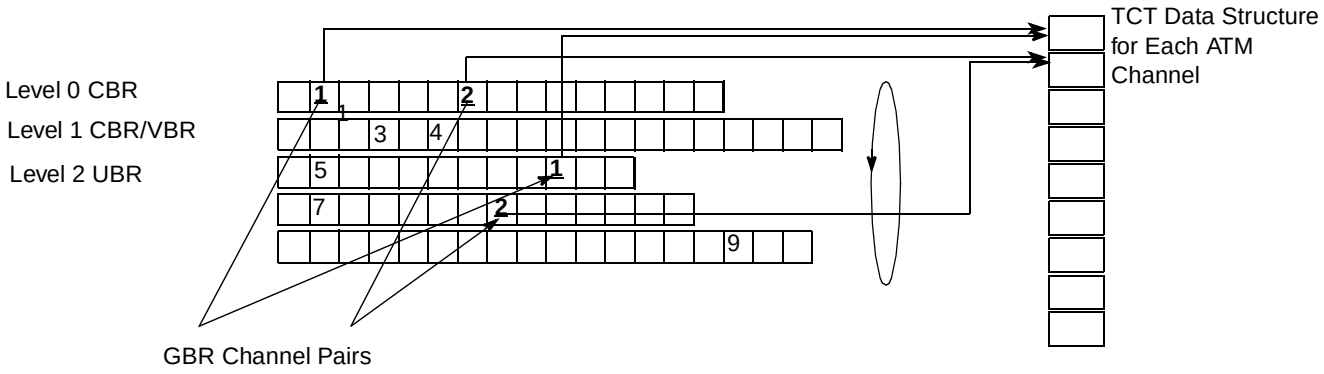
30.2.9 GBR Guaranteed Bit Rate

In systems where RT traffic and control traffic share the same bandwidth there is a need to multiplex these various type of traffic (data delay sensitive, data delay non sensitive and control/management flow) assuring each type gets proper priority and bandwidth. The goals set are:

- Guarantee minimum bit rate for control/management flow.
- Guarantee bit rate and latency for RT traffic.

In cases where there is an excess bandwidth available, distribute it to the rest of the flows following the general traffic priority rules, like delay sensitive data non delay sensitive data and control flow.

In the Guaranteed Bit Rate (GBR) mode the same channel can be scheduled in CBR & UBR priority levels concurrently, having different contracts on each priority and transmitting common data.



30.2.10 Prioritized UBR+ Mode

In systems with more than one APC priority level used by UBR+, the highest priority UBR+ level actually can take over all the capacity from the lower priority UBR+ levels, even if an MCR parameter is specified for lower priority levels. This is because usually the user use oversubscribing when he initialize the UBR+ PCR parameter.

This requires that guaranteed traffic (MCR) and non-guaranteed traffic of UBR+ VCs are required to always be in the same priority level, instead of having the guaranteed traffic (MCR) components of all UBR+ connections in highest priorities of the UBR+ levels, and then the non-guaranteed traffic would be scheduled at the lowest priority levels.

The prioritized UBR+ mechanism is a method for enabling the possibility to use UBR+ in different priority levels. This UBR+ channels get service according to two factors:

- The difference between the real-time pointer and service pointer for this level against the minimum delay allowed (MDA) threshold value set for this level.
- The priority level.

The user must configure the APC priority levels so that the low priority levels up to the last priority level contain UBR+ channels. See Figure 30-5, for an example APC with 6 priority levels, of which 4 are UBR+.

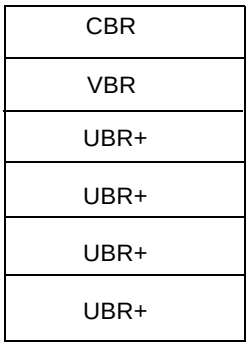


Figure 30-5. UBR+ Priority Level Example

The user must initialize the UBR+ priority decision table (see Figure 30-46) before enabling the UBR+ prioritized mode. This table is located at address Scheduler_Parameter_Table [AFP_LAST]+8. See Figure 30-6, for an example UBR+ priority decision table that contains four entries. The UBR+ prioritized mode is enabled by setting the PUBR+ bit in the Scheduler_Parameter_Table[Scheduler_MODE] entry.

NOTE

When there is only 1 UBR+ priority level, the UBR+ prioritized mode must be disabled.

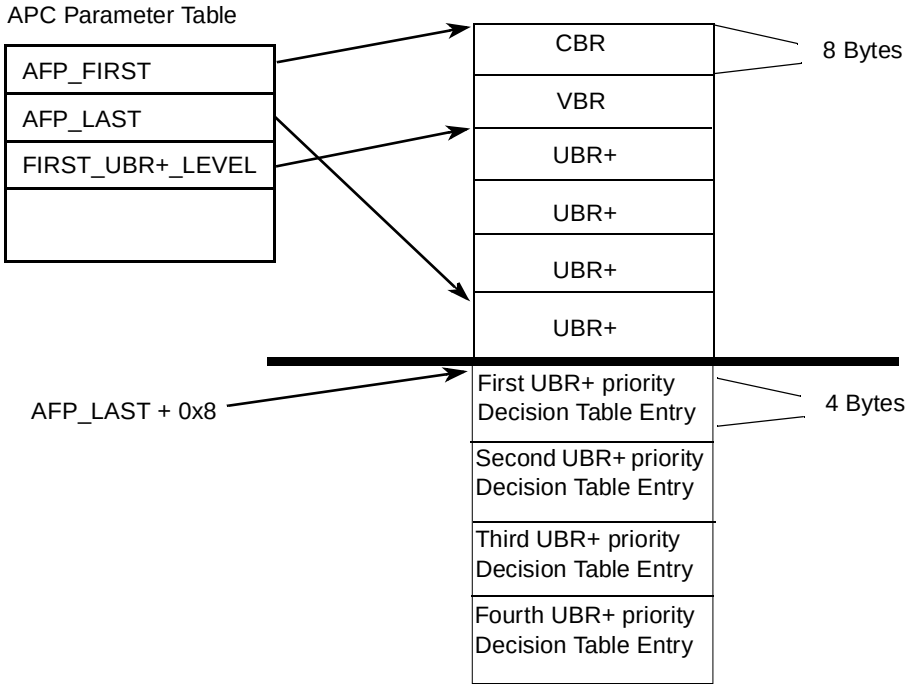


Figure 30-6. UBR+ Priority Table Example

30.2.11 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 30.3.6, "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 30.4.1, "ATM Commands."

The real-time channels, CBR and VBR-RT, should be inserted in APC_LEVEL1; non-real-time channels, VBR-NRT, and UBR should be inserted in lower priority levels.

30.2.12 GCRA Scheduler

The Generic Cell Rate Algorithm scheduler mechanism is a non-work conserving ATM scheduler, similar to the traditional APC. This CGRA scheduler method is ideal for systems with a large number of PHYs per UCC, the number of channels per PHY is relatively small (up to 64 channels or for a smaller memory footprint 32 channels), and the variance between the channels rates is big. This is normally the case in DSLAM applications. The big advantages of the CGRA scheduler method compared to the traditional APC algorithm are the smaller internal RAM usage and the improvement in performance.

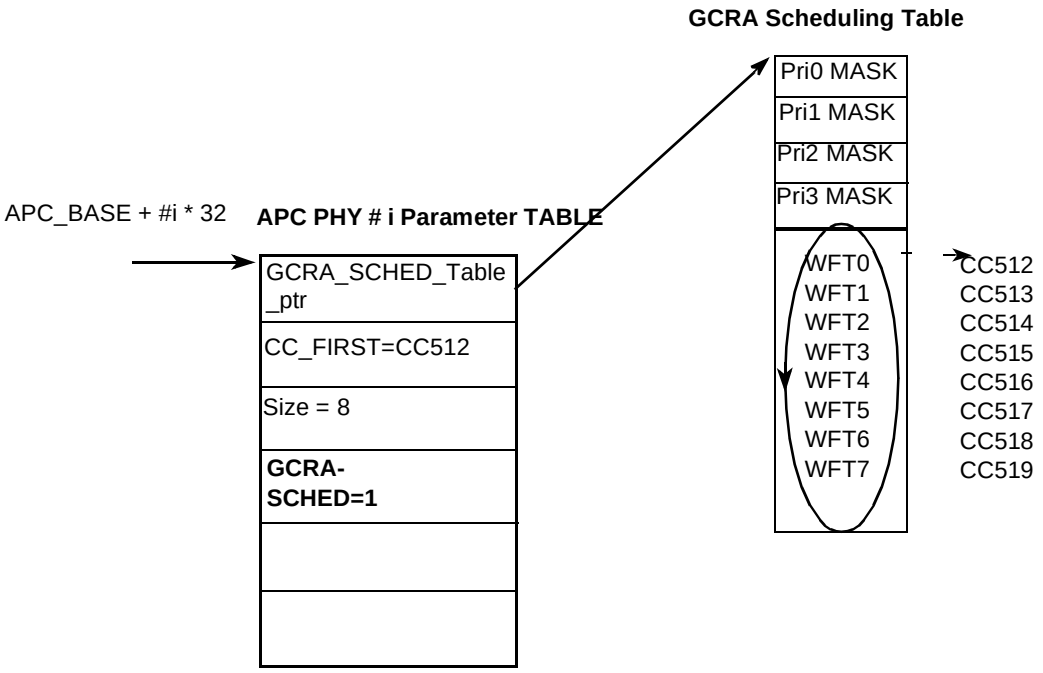


Figure 30-7. GCRA Scheduling structure

Figure 30-7 describes an example of GCRA scheduler. Each PHY which is programmed to work in GCRA scheduler mode, points to a GCRA scheduling table. This table contains the timing information (WFT) for a GCRA algorithm for each channel. The algorithm checks eligibility of all the channels to transmit. The channel which has the highest credit at a given time is selected for transmission. Once transmitting a cell the, GCRA is updated and the channel is rescheduled for transmission according to the parameters

specified. Up to four priority levels are available for each PHY. Each priority has its own Priority MASK register, which identifies which channel out of the 64 available channels belongs to the priority level. The search in each priority is performed based on the bits which are set in the mask register. This MASK is also useful for polling mode. The QUICC Engine block clears this bit when there is nothing to transmit from the channel code (TCT[AVCOFF]=1), and the core set this bit, by issuing an ATM TRANSMIT COMMAND, when the next buffers are ready for transmission. In this way the QUICC Engine block is better utilized since channels with an empty queues are not taking part in the selection process.

For more information about the GCRA scheduler structures See Section 30.3.7, “GCRA Scheduler Structures.”

30.2.12.1 GCRA Scheduler Rate Programming

The values which are programmed in the TCT/TCTE are different then the standard APC and they have a meaning of timing. Time is measured by the QUICC Engine time stamp and a normalized time unit called “Increment”-INC is defined such that it reflect the time for transmitting a single cell at a specified rate. The formula for calculating the value of the INC parameter is described below:

$$\mathbf{INC = 424/(TSR\ Time\ Unit[Sec] \times Channel\ Rate[bps])}$$

NOTE

Programming of CETSCR time unit should be such that each tick is smaller than the cell transmission time of the fastest channel in the APC tables.

When programming a rate value in an ATM channel the values programmed are no longer specified in APC slot but in normalized time units so that PCR->1/PCR, SCR->1/SCR, OOBR-> 1/OOBR and MCR->1/MCR.

Example 30-1. GCRA Scheduler Programming:

Assuming the highest rate is CH #5, rate= 12Mbps and the lowest rate is CH #6, rate = 32Kbps. If using Time unit which is at the highest rate then value for CH #5 INC is 1, and the value for CH #6 INC is 12M/32K= 375. The Time stamp time unit should be programmed to:

Time Unit=424/12M=35.3 uS.

In case the Time Unit is 1usec., then we get:

$INC\ (CH5)=424/(1 \times 10^{-6} \times 12 \times 10^{6})=35.3 => PCR=0x23,\ PCR\ FRAC=0x55$

$INC\ (CH6)=424/(1 \times 10^{-6} \times 32 \times 10^{3})==>PCR=0x33B5\ PCR\ FRAC=0x80$

30.2.12.2 Dynamic Adding and Removing Channels

Removing a channel from the GCRA scheduling table is done by setting the TCT[STPT] bit or by setting the TCT[AVCOFF] bit. In both cases, the QUICC Engine block is responsible to clear the corresponding bit in the Pri_MASK entry in the GCRA scheduler PHY Parameter Table. The host must not change this bit by itself.

To add a channel to the GCRA scheduling table, the user must issue an ATM TRANSMIT command, see Section 30.4.1, “ATM Commands.” This command actually sets the corresponding bit in the Pri_MASK field.

30.2.13 Hierarchical Scheduling

30.2.13.1 Frame Based Scheduling

In some applications the number of VCs is limited while there is still a need to maintain a few types of QOS for each VC. The standard definition of an ATM channel in the QUICC Engine block can support a single queue for transmission. It is not allowed to have several ATM channels having the same VPI/VCI, especially when running AAL5 traffic, since it could cause interleaving of data, and for this reason it is required to establish an hierarchy for transmission for each ATM channel. A channel is scheduled according to predefined contract using APC or GCRA scheduler. This channel can support up to eight queues for transmission, all queues have the same VPI/VCI. Selection is done by a WFQ algorithm. Once a queue is selected, the transmission from this queue continues until a complete frame is transmitted. See complete programming model in Section 30.3.8, “Hierarchical Scheduling.”

30.2.13.2 Hierarchical Cell Based Scheduling

The QUICC Engine block allows hierarchical scheduling where each queue in the second hierarchy has a different VPI/VCI. There is no potential interleaving problem between different AAL5 frames. The WFQ selection is performed a single ATM cell is transmitted. This functionality is described in Section 30.3.8, “Hierarchical Scheduling.”

30.2.14 VCI/VPI Address Lookup Mechanism

The QUICC Engine block supports three ways for address look up of incoming cells:

- Address compression see Section 30.2.14.1, “Address Compression”
- Mini-CAM address look-up see Section 30.2.14.2, “Mini-CAM Address Look-Up”
- Channel Code extracted from User Defined Cell header. see Section 30.2.16.1, “Channel Code Extracted from UEAD_OFFSET”

Writing to GMODE[ALM] (address-lookup-mechanism bits) in the parameter RAM selects the mode of operation. These three modes are described in the following sections.

30.2.14.1 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 Utopia bus ID, PHY address and the 12-bit virtual path (VP) 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 miss-inserted cell.

During the VP-level translation, VP_MASK in the ATM parameter RAM compresses an incoming cell’s Utopia ID, 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 is recommended to reside in the multi-user 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 30-8 shows an example of address compression.

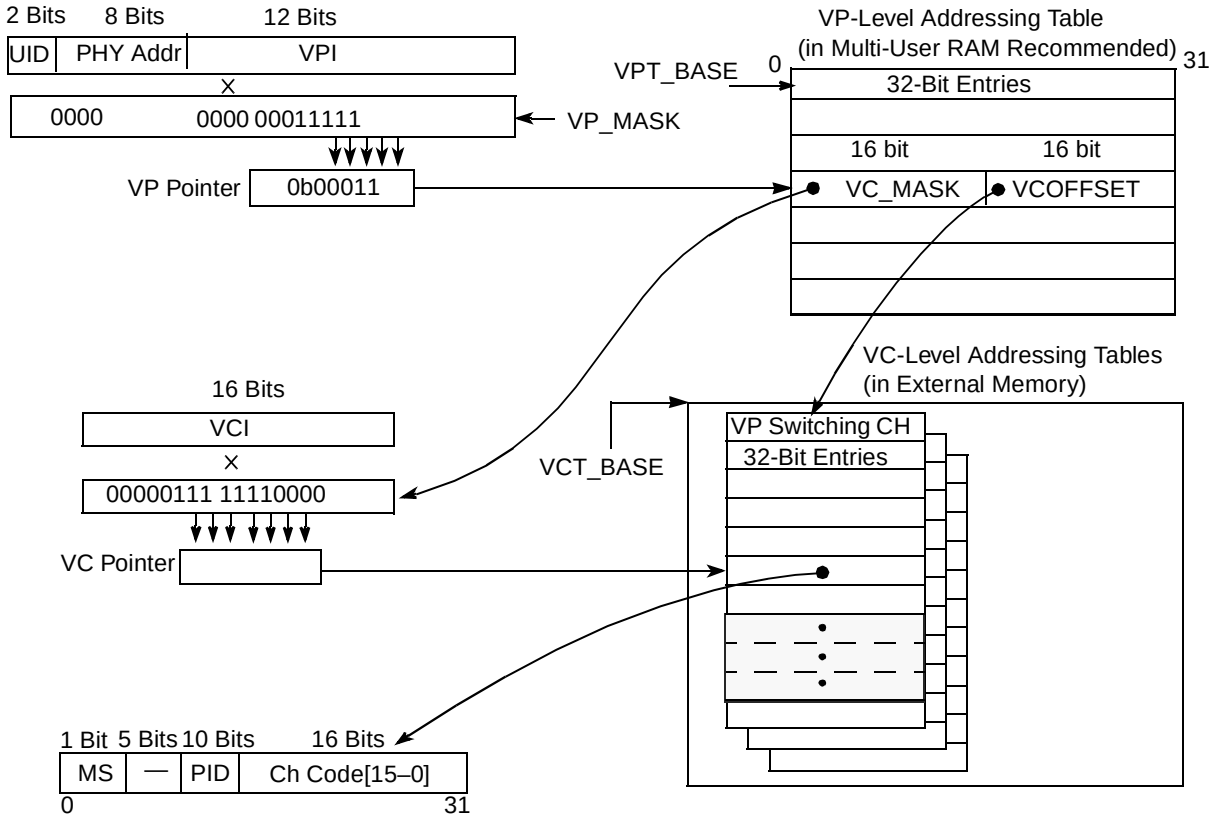


Figure 30-8. Address Compression Mechanism

Figure 30-8 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 30-2.

Table 30-2. Field Descriptions for Address Compression

Bits	Name	Description
—	UID	Utopia ID. Indicates the UTOPIA bus on which the ATM cell has been received.
—	PHY Addr	In multiple PHY mode, this field contains the PHY address. PHY ID should be treated as follows: 5 LSBs represent the 5 address lines of the UTOPIA interface and 3 MSBs represent the device ID.
—	GFC+VPI, VCI	The GFC, VPI, and VCI of the current channel.

Table 30-2. Field Descriptions for Address Compression (continued)

Bits	Name	Description
0	MS	Match status. 0 Match was found. 1 Match was not found.
1–5	—	Reserved, should be cleared.
6–15	PID	UPC ID. The bits are used to determine which UPC table is used for policing of this channel. 0 - UPC is disabled. 1-31 - UPC tables reside in the internal multi-user RAM starting from address INT_UPC_BASE. 32-1023 - UPC tables reside in external memory starting from address EXT_UPC_BASE.
16–31	Ch Code	Pointer to internal or external connection table.

30.2.14.1.1 VP-Level Address Compression Table (VPLT)

The size of the VP-level table depends on the number of mask bits in VP_MASK. This level mask can contain up to 22 bits. It contains the UTOPIA ID which is 2 bits, PHY address which has 8 bits assigned, and 12 bits for the GFC+VPI of the incoming ATM cells. 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 30-9 gives the general formula for determining VCOFFSET.

General formula: $VCOFFSET_{(n+1)} = VCOFFSET_n + 2^{(\text{number of ones in VC\_MASKn})}$

Figure 30-9. General VCOFFSET Formula for Contiguous VCLTs

Table 30-3 shows example VCOFFSET calculations for a VP-level table with four entries.

Table 30-3. 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

IMPORTANT: Since the offset field in each VPLT entry pointing to a VCLT table is only 16-bits, the maximum number of VC entries is limited to 64K VC's. In some configuration this might not be enough.

For example: A multi PHY system with 128 PHYs where the number of the mask bit on the VP is 6 bits. This takes 7+6=13 bits for the VP level. There are 2^{13} VP level entries. Each entry should have its offset to the VC level table. Since the VCT Base is common to all we have, on the average, a maximum of 2^3= 8 entries per VC table which might not be enough.

In order to solve this, there is an option to access the VC level with a different scale factor which span the memory and the tables could be bigger. Instead of accessing the table by:

VCLT = VC_BASE + VCOFFSET × 4, it uses a different scale factor (VCLT_SF) which is programmable. The address now is: VCLT = VC_BASE + VCOFFSET × $2^{(VCLT\_SF+2)}$. If the scale factor is not zero the offset calculation for each VCLT is different and has

The QUICC Engine block can check that all unallocated bits of the UID+ PHY + VPI are 0 by setting GMODE[CUAB] (check unallocated bits) in the parameter RAM. In case there are some un-allocated bit the QUICC Engine block will treat the incoming cell as a miss-inserted cell and discard the cell. Table 30-4 gives an example of VP-level table entry address calculation.

Table 30-4. VP-Level Table Entry Address Calculation Example

VPT_BASE	VP-Level Table Size	VP_MASK	UID+Phy+VPI	VP Pointer	VP Entry Address
0x0024_0000	64 entries	0x0237	0x0011	0x09	VP Base = 0x240000 0x09 x 4 = 0x000024 0x240024

Figure 30-10 shows the VP pointer address compression from Table 30-4.

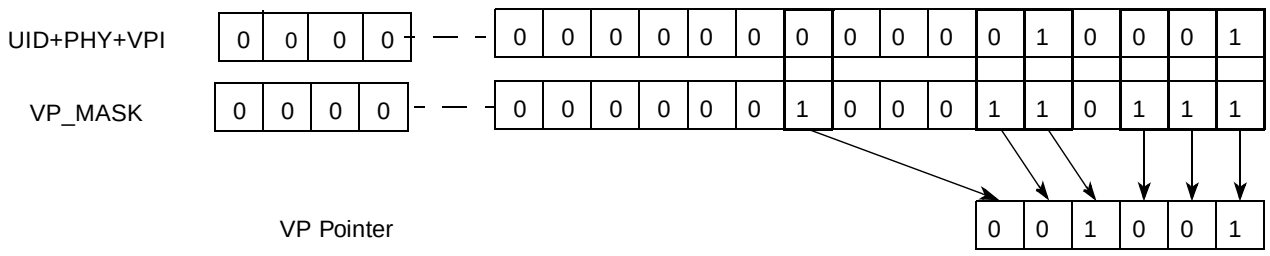


Figure 30-10. VP Pointer Address Compression

30.2.14.1.2 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. By default, the address of an entry in this table is VCT_BASE + VCOFFSET × 4 + VCpointer × 4, but it could be programmed to a different scale factor in order to span a bigger memory space when more VC are needed.

In the VC level the QUICC Engine block 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 miss-inserted cell. or alternatively, based on a mode in the UCC MODES, activate a **VP switching** flow where all incoming traffic is directed to a dedicated ATM VC. In this case the default ATM channel is located at the first VC-level table at offset 0x0.

An example of VC-level table entry address calculation is shown in Table 30-5. Note that VCOFFSET is assumed to be 0x100 for this example.

Table 30-5. 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 30-11 shows the VC pointer address compression from Table 30-5.

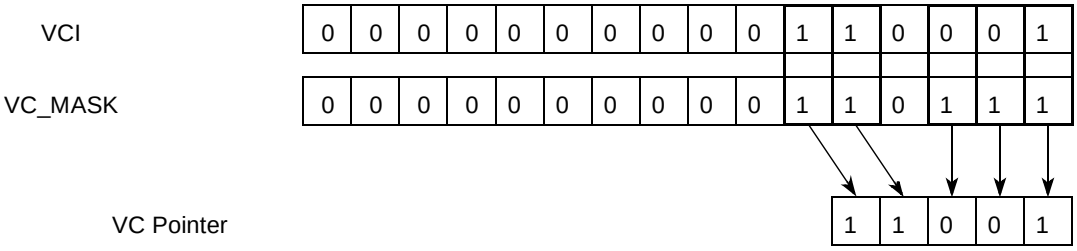


Figure 30-11. VC Pointer Address Compression

30.2.14.1.3 Dynamic Change of Address Compression Tables

It is possible to change the lookup table dynamically to get an updated mapping of ATM channels without the need to stop operation of the devices. See detailed description at Section 30.3.2.8, “Dynamic Address Compression Table change”.

30.2.14.2 Mini-CAM Address Look-Up

This mode of operation is suitable for systems where the number of VPI/VCI is relatively small. It is efficient due to the fact all the look-up is performed in internal MURAM and there is no access to external memory through DMA operations. The MURAM consumption is reduced significantly if the VPI/VCI are repetitive on the PHYs. The principal of this look-up is to have a match on a [GFC]VPI/VCI on a mini-CAM table located in the MURAM.

For each UCC there is a Mini CAM parameter table containing common parameters for this UCC. It replaces the Dynamic Address Look-up table. It contains a base address for the PHY table and for the array of mini-CAM tables for each PHY as indexed from the PHY table. Figure 30-12 illustrates the operation of this look-up

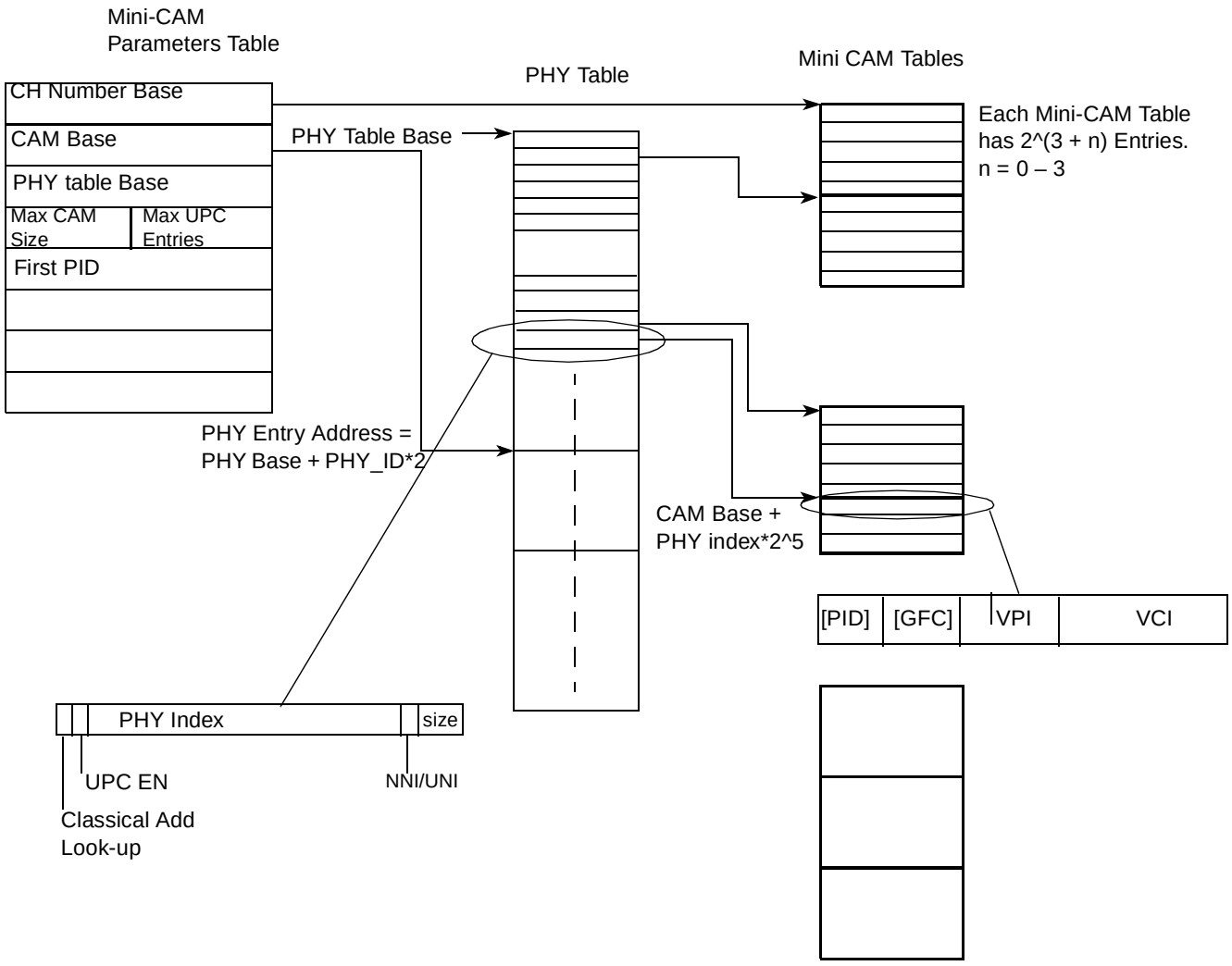


Figure 30-12. Mini-CAM Address Look-Up

The data structures used in this address look-up mode are described in the following paragraphs:

30.2.14.2.1 Mini CAM Parameter Table

When working in Mini CAM look-up this table replaces the dynamic address look-up table. It is described in details in Table 30-27. It contains the following fields:

Base_Channel#—This is the base channel number for calculations of the ATM channel code.

Mini-CAM_base—Base offset in the MURAM to the mini CAM tables

PHY_table_base—Base offset in the MURAM offset, to the PHY table

Max_MC_size—Number of entries in the largest mini CAM table in the system. Size of this table could be one of the following: 8,16,32,64. The value programmed in Max_MC_size is the power of 2 of the size so values are 3,4,5,6 respectively.

Max_UPC_Entries—If Policing is enabled in this PHY up to 15 PID (Policer ID) are available for each PHY. This value is maximum value of UPCs used for any of the PHYs. This value should be rounded to the next power of 2. The value programmed in Max_UPC_Entries is the power of 2 of the size (for example if the maximum number of UPCs used per PHY is 5 set this value to 3)

First_PID—The first Policer ID used in the system. Any of the UPCs used is calculated based on this number.

30.2.14.2.2 PHY Table

Each entry in the table is indexed by the PHY ID and consists of two bytes:

Bits 14-15 determines the number of entries in the mini-CAM table (0b00=8, 0b01=16, 0b10=32, 0b11=64) The size of the mini-CAM table should be set in such a way that it will contain all the required VC's and minimizes the size of the table. In this way MURAM usage is smaller and the performance is increased.

Bits 13 defines UNI/NNI mode (such as in NNI mode keep GFC and in UNI clear GFC for address check in lookup)

Bits 2-12 defines index into the Mini-CAM_base using 2^(5)*index as the mini-CAM address for this PHY.

Bit 1 UPC enabled for this PHY

Bit 0 - use classical lookup method and NOT the mini-CAM for this PHY. Use the Address look-up table as the table for the address compression parameters.

30.2.14.2.3 Mini-CAM Table

This is the Mini-CAM table for each PHY. It contains the actual GFC/VPI/VCI which are searched for this PHY. In cases where some PHYs uses identical GFC/VPI/VCI the index value in the PHY table entries for these PHY should be identical so that the mini-CAM tables are overlapping. The actual ATM channel number is different as it is based on the following calculation:

Channel# = Base_Channel# + PHY_ID * Max_MC_size + Mini-CAM index

If Policing is enabled the four MSB of each entry are used to identify a Policer ID to be used for this VPI/VCI. A zero in these bits indicate No Policer is enabled

PID# = First_PID + PHY_ID * Max_UPC_Entries + mini-CAM entry[PID]

30.2.14.2.4 Mini-CAM Lookup Process

Figure 30-13 is a schematic flow of the look-up process.

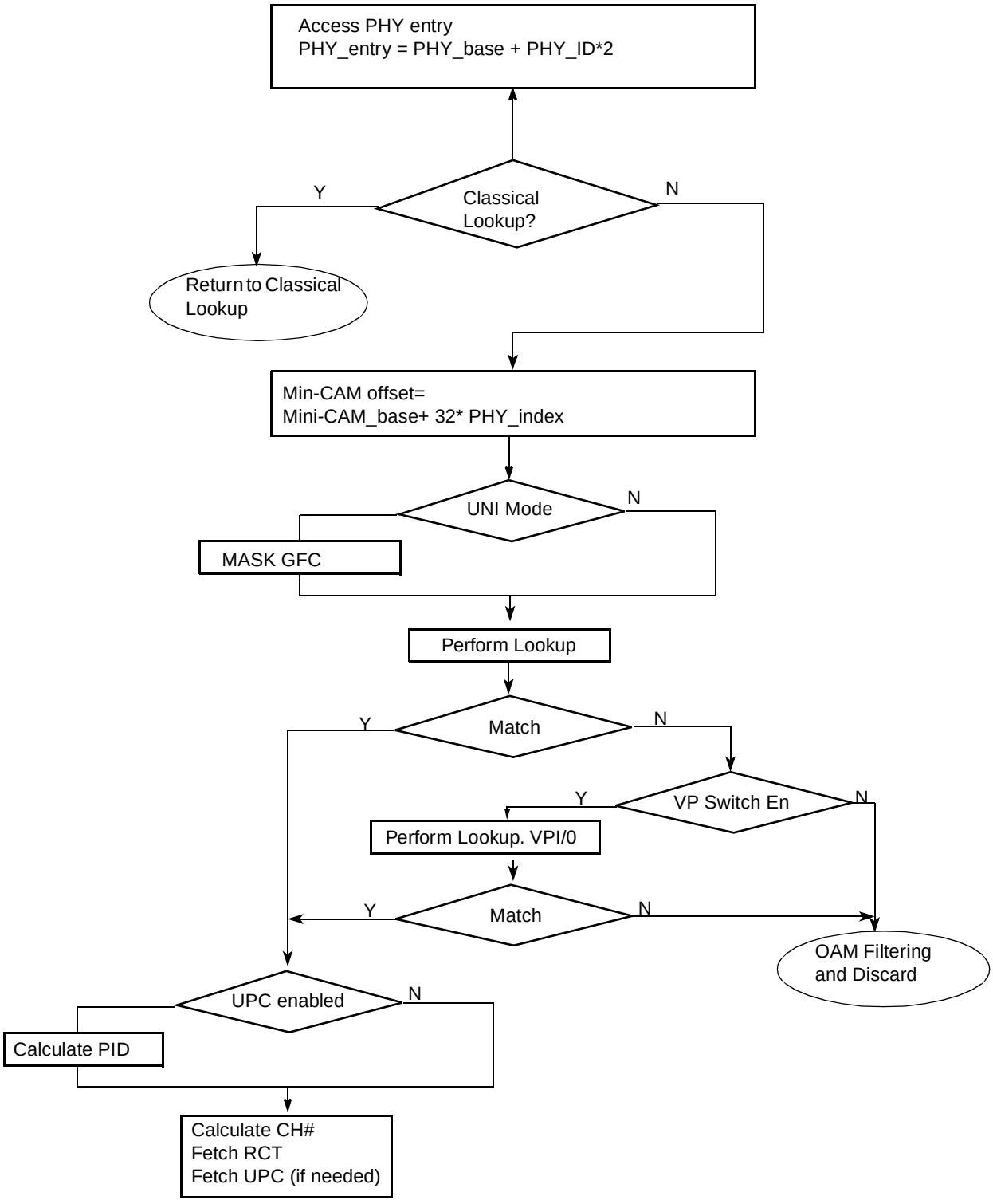


Figure 30-13. Mini-CAM Look-Up Flow

30.2.15 Miss-Inserted 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 30.2.21, “ATM Layer Statistics.”

30.2.16 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 QUICC Engine block 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 UPSMR, see Section 31.6.5.1, “UPDCx in ATM Protocol.” As shown in Figure 30-14, the extra header size can vary between 1 to 12 bytes (byte resolution) and the HEC octet is optional.

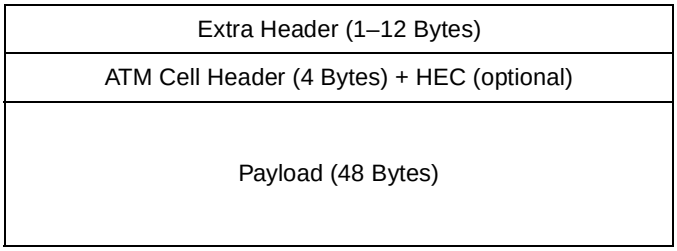


Figure 30-14. Format of User-Defined Cells

For AAL5 the extra header is taken from the Rx and Tx BDs. 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.

30.2.16.1 Channel Code Extracted from UEAD_OFFSET

When working in User Defined Cells mode enabled, it is sometimes more efficient to disable the address look-up mechanism and to read the Channel Code (16 bits) directly from the Extra Header; this mode is enabled by programming GMODE[ALM]=10 (see section Section 30.3.2.5, “Global Mode Entry (GMODE)”). When using this mode, the UTOPIA port must set to User Define Cells (UDC) mode and include at least 2 bytes of Extra Header. CH_CODE_OFFSET entry in the parameter RAM specified the offset of the CH_CODE entry from the beginning of the Extra Header field. It should be an even address.

The CH_CODE entry can be located in 6 locations within the Extra Header. For each location CH_CODE_OFFSET should be set as shown in Table 30-6.

Table 30-6. CH_CODE Location In Extra Header

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
CHANNEL_CODE_OFFSET=2	CHANNEL_CODE_OFFSET=0
CHANNEL_CODE_OFFSET=6	CHANNEL_CODE_OFFSET=4
CHANNEL_CODE_OFFSET=10	CHANNEL_CODE_OFFSET=8

30.2.16.1.1 Channel Code Protection Mode

In “Read Channel code from Extra Header” mode, there is an optional protection mechanism, to prevent handling of cells with corrupted channel code. If enabled, in GMODE[ALM]=11 (see section Section 30.3.2.5, “Global Mode Entry (GMODE)”), the QUICC Engine block expects two occurrences of the same channel code appearing in the UDH (extra header) one after the other. The QUICC Engine block compares the two values, and processes the cell only if they are identical. Otherwise, the cell is treated as a miss-inserted cell—same treatment as in the case where MS=1 in the previous address lookup mechanisms. See section Section 30.2.15, “Miss-Inserted Cells.” Note that if the protection mode is enabled, the value of CHANNEL_CODE_OFFSET cannot exceed 8, since we need 4 bytes for the 2 occurrences of the channel code (max UDH size is 12 bytes). The CHANNEL_CODE_OFFSET points to the first occurrence of the channel code in the UDH (extra header).

30.2.17 Receive Raw Cell Queue

Each UCC has a raw cell queue designated for OAM handling. This queue is managed by a designated ATM Channel per UCC (RCT) which is called the raw cell queue. The user should program this channel to operate in AAL0 mode. This channel’s parameters are pointed by **OAM CH RCT PTR** which resides in parameter page of each UCC. For details see Table 30-18.

NOTE
On Backward Compatibility

Allocating AAL0 RCT for raw cell queue is no longer by allocating channel #1 as the channel for raw cell queue. Channel #1 shouldn’t be assigned in the system at all!

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 QUICC Engine block sets RxBD[OAM]. The ALL0 BD specifies the channel code associated with the current OAM cell. Interrupt entry for ALL the OAM cells will contain the CC - Channel code #1 as the cause for the interrupt. The OAM buffer will contain the Time stamp value on which it has been received.

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). 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 30-15 shows a flowchart of the ATM cell flow.

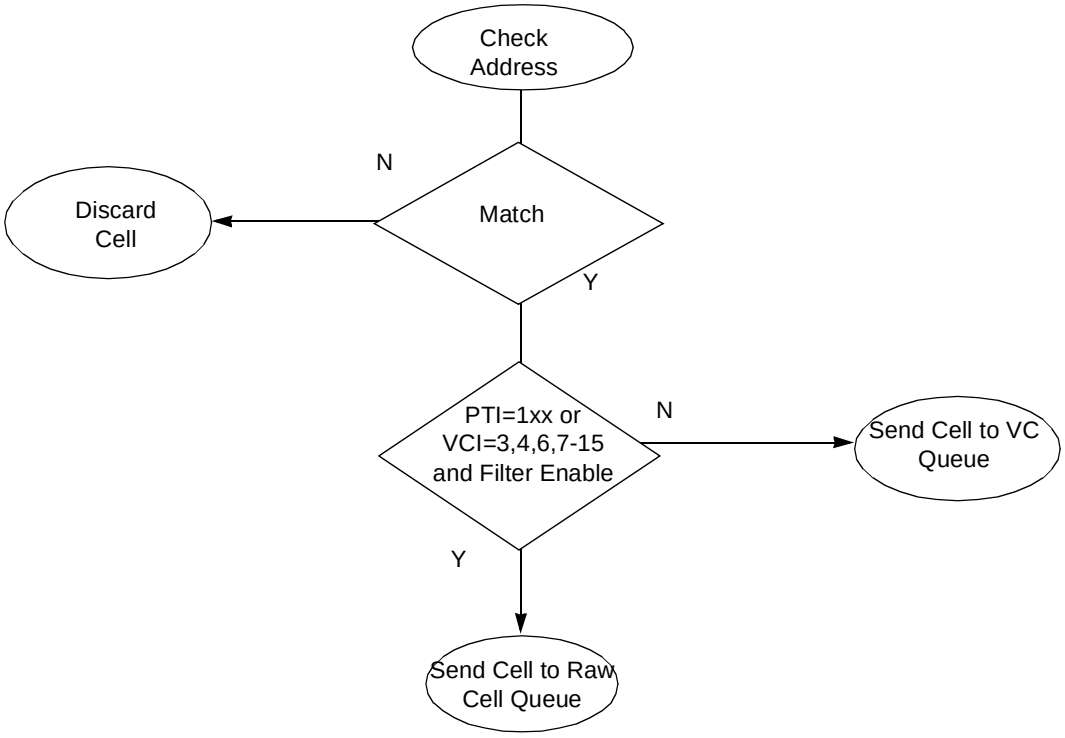


Figure 30-15. ATM Address Recognition Flowchart

NOTE

Even reserved VCI channels should appear in the CAM or address compression tables; otherwise, a cell on a reserved channel is considered miss-inserted.

30.2.18 OAM Support

This section describes the QUICC Engine block'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 QUICC Engine block automatically inserts forward monitoring cells (FMC) and generates backward-reporting cells (BRC) as recommended by ITU I.610.

30.2.18.1 ATM-Layer OAM Definitions

Table 30-7 lists pre-assigned header values at the user-network interface (UNI).

Table 30-7. 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 30-8 lists pre-assigned header values at the network-node interface (NNI).

Table 30-8. 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

30.2.18.2 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.

30.2.18.3 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.

30.2.18.4 Receiving OAM F4 or F5 Cells

OAM F4/F5 flow cells are received using the raw cell queue, described in Section 30.2.17, “Receive Raw Cell Queue.” An F4/F5 OAM cell which does not appear in the mini-CAM or address compression tables is considered a miss-inserted cell.

30.2.18.5 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 30.3.4.2.2, “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.

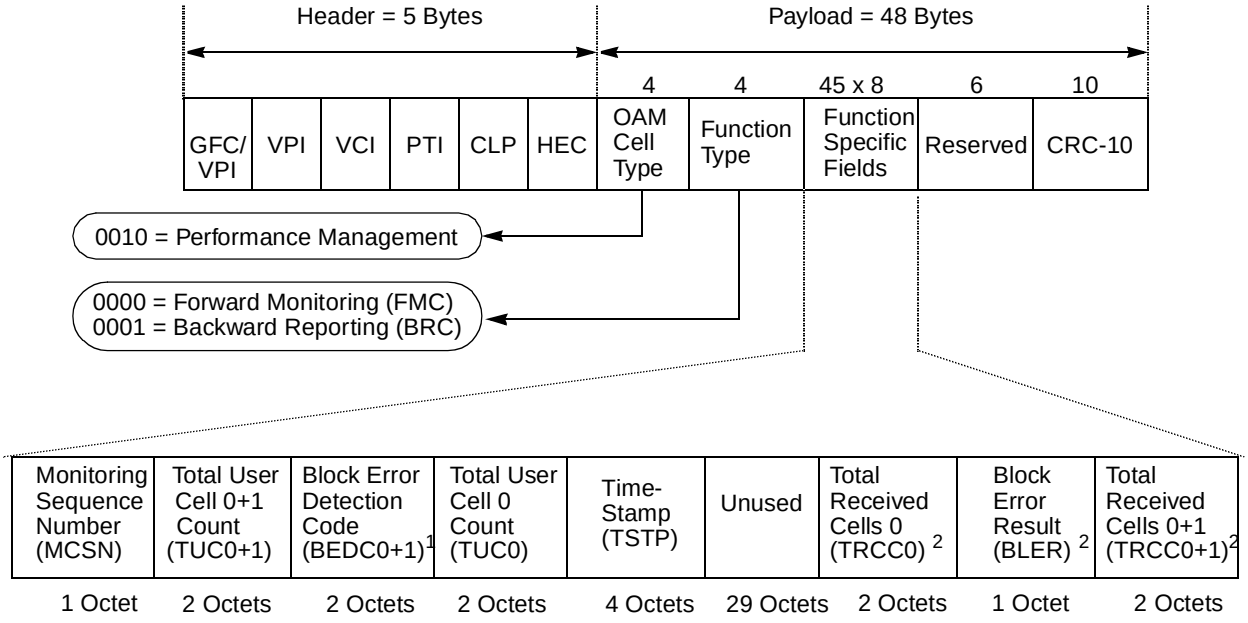
A series of OAM cells can be sent using one ATM TRANSMIT command by creating a table of AAL0 TxBDs. If the channel’s TCT[AVCF] (auto VC off) is set, the transmitter automatically removes it from the APC (that is, it does not generate periodic transmit requests for this channel after all AAL0 BDs are processed).

30.2.19 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 QUICC Engine block 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 30-16 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 30-16. Performance Monitoring Cell Structure (FMCs and BRCs)

Table 30-9 describes performance monitoring cell fields.

Table 30-9. 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

30.2.19.1 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 30.3.5, "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.

30.2.19.2 PM Block Monitoring

PM block monitoring is done by the receiver. After initialization (see Section 30.2.19.1, "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 QUICC Engine block 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.

30.2.19.3 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 QUICC Engine block inserts an FMC into the user cell stream. The QUICC Engine block 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 QUICC Engine time stamp timer, see Section 19.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)."

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 30.3.5, "OAM Performance Monitoring Tables."

In Figure 30-17, 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.

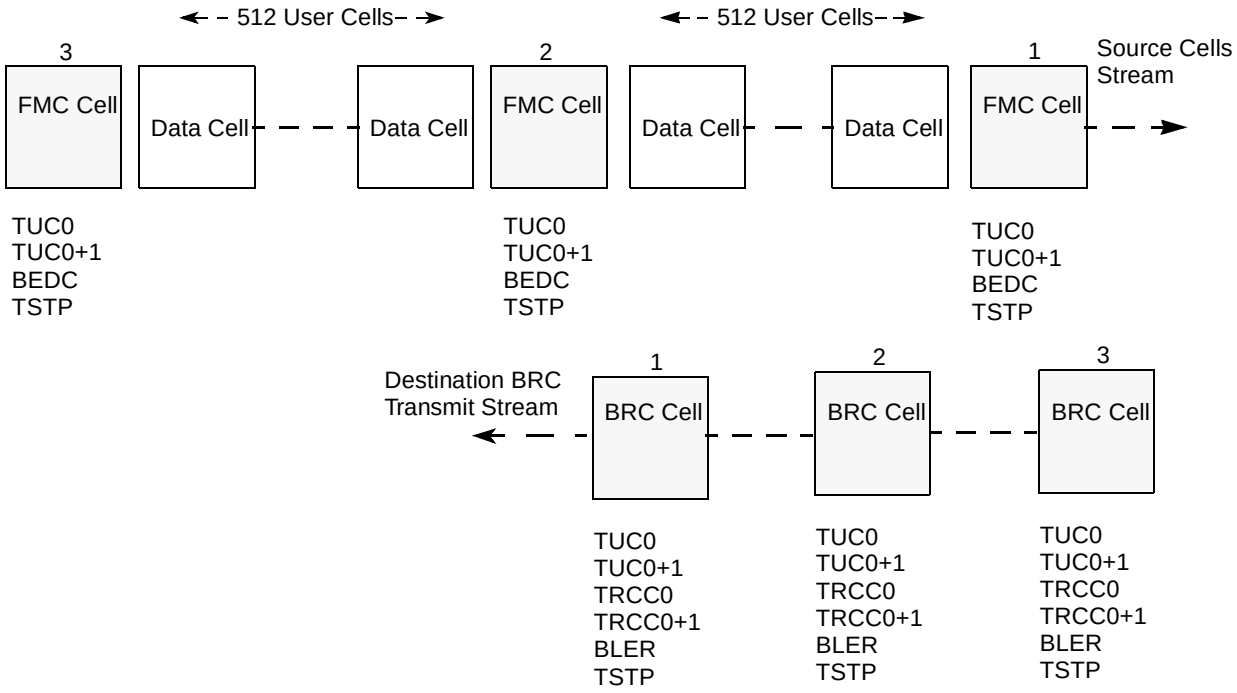


Figure 30-17. FMC, BRC Insertion

30.2.19.4 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 miss-inserted cells of this block test.
- When Nt = Nr, no cells are lost or miss-inserted.

30.2.20 UPC

The User Policer Controller (UPC) implements ATM forum's TM4.1 Virtual Scheduling algorithm. Providing a dual leaky bucket policing scheme, the UPC can detect cells that violates the traffic agreement (Non conformed cells) and optionally tag or discard them from the cell flow.

A Leaky Bucket parameter structure is defined either on a per connection (VCC) basis, per VPC basis, or per any arbitrary group of VCC's. Any address lookup resolution PID field output contains a pointer to a Dual Leaky bucket parameter structure (UPC parameter structure), see Table 30-5 and Table 30-4 for more description about the PID.

When policing has to be done for a group of VCC's (VCC's Bundle), then all the address lookup entries belonging to the same group have to point to the same UPC parameter structure. In this case the UPC controls the traffic stream generated by the unification of those VCC's.

Both leaky buckets independently perform the ATM Forum's TM4.1 **GCRA(I,L)** algorithm, where I and L express Bucket Increment and Bucket Limit respectively. In the UPC programming model, I is defined by **B1I** and **B2I** (Bucket 1 increment and Bucket 2 increment respectively), which is represented by an integer part, **B1II** (**B2II**), and a fraction part, **B1IF** (**B2IF**).

B1I (**B2I**) and **B1Lim** (**B2Lim**) represent the space between cells arrival times and the limits for leaky bucket 1 (2). Both have to be initialized by the user. The time units for both, the bucket increments and the bucket limits are identical to one-count time interval of the CETSCR - the QUICC Engine Time Stamp Control Register, see Section 19.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)."

The Theoretical arrival times for bucket 1 and bucket 2 (**B1T** and **B2T**) are internal variables which are maintained by the Leaky Bucket mechanism.

Each of the leaky buckets can be programmed to test GCRA conformance of CLP0, CLP1 or CLP0+1 cell streams, by setting the appropriate value to **LBF** (Leaky Bucket Filter) field (LBF1 for Leaky Bucket #1 and LBF2 for Leaky Bucket #2), see Table 30-10. Both LBF1 and LBF2 have to be initialized by the user.

Table 30-10. LBF - The Leaky Bucket Filter

Cell type monitored	LBF coding
Disable policing	000
GCRA CLP0 cell policing. CLP1 cells are bypassing this bucket	001
GCRA CLP1 Cell Policing. CLP0 cells are bypassing this bucket	010
GCRA CLP0+1 Policing. All cells are tested by this bucket	011
F-GCRA Policing. Can be applied only on bucket1, when using GFR mode. Only First cell of each packet with CLP=0 is tested for conformance. All the rest of the cells gets the same conformance result as first cell. If CLP=1 in the first cell the DM bit determines the action, which will preformed on the whole packet.	100
Reserved	101
Reserved	110

NOTE

OAM cells bypass the UPC (not checked by the UPC).

Only bucket1 can be programmed to test F-GCRA.

30.2.20.1 UPC Programming

Before enabling the UPC for any channel, the CETSCR must be initialized. The CETSR defines the time unit which is common to all leaky buckets and timestamps in the QUICC Engine block. Before a specific UPC is enabled, the following fields must be programmed:

- UPC Parameter Table fields (see Table)
- UPC Table fields (see Table 30-59. The CETSCR for the UPC is always CETSCR1.

The formula for the decimal representation of the bucket increment is:

```
A)       BI = CellSize/(BitRate x TimeUnit)
```

- BI is the decimal expression for the bucket increment.
- CellSize is the size of a cell on the physical media considered (in units of bits)
- TimeUnit is the time interval of one count of CETSR. Since CETSCR is global for all connections, TimeUnit has to be programed to be not bigger than the time required for one cell on the fastest connection.

After BI computation, it must be converted to hexadecimal fixed-point representation.

In the following example, the leaky bucket must be programmed for a connection with a rate of 16 Kbps, where the fastest connection in the system is 155 Mbps. We find the value of BI, assuming a cell size of 53 bytes:

```
B) TimeUnit <= 53x8/155Mbps = 2.735us
```

CETSCR is programmed to 2.7us. Substituting the value in (A) we get:

```
BI = (53x8)/(16kbps x 2.7us) = 9814.8148
```

Converting it to hexadecimal yields: BI = 0x2656.D09, which implies the following programming:

```
B1II[0:11] = 265, B1II[12:15] = 6, B1IF[0:11] = D09
```

The smallest data rate given by the substitution of BI = 64 K in (A), is 2400bps. The resolution is at least 0.02 %. The formula for the bucket limit is:

```
C) BLim = CDVT/TimeUnit
```

- BL is the decimal expression for the bucket's Limit.
- TimeUnit is the time interval of one count of CETSR. For details on TimeUnit, refer to the explanation of BI programming.

NOTE

The most significant bit of the BL value must be cleared.

Following the calculation in (C), the result is rounded to the closest integer and converted to a hexadecimal representation. The smallest value of L is 1 TimeUnit (2.7 us in the preceding example). The highest value is 190 min.

Example 30-2. Using PCR, CDVT

- Highest Frequency: 16 Mbps
- PCR = 1536 Kbps
- CDVT = 0.5 sec

```
TimeUnit <= 53x8/16Mbps = 26.5us.
```

This value of `TimeUnit` is larger than the highest possible value in CETSR, so we program CETSR to `TimeUnit` = 5us.

```
BI = (53x8)/(1536kbps x 5us) = 55.2083 -> BI= H37.355 -> BII=0037, BIF=355

BL = 0.5/5us = 100,000 -> BL=0x186A0
```

30.2.20.2 UPC Operation Modes

30.2.20.2.1 Cell Base UPC

When UPC Table[UPCM]=01, the UPC works in cell base mode. In this mode, the buckets can be configured to one of the GCRA mode. Cell which fails in the bucket1/2 test will be dropped or tagged according to DM1/2. Tagging is done by setting PNC bit in the RxBD. The host software can tag outgoing cells according to PNC.

30.2.20.2.2 Frame Awareness UPC

When UPC Table[UPCM]=10, the UPC use frame awareness mode. The buckets can be configured to one of the GCRA mode or to F-GCRA which is explained in Section , “GFR Frame-Eligibility (F-GCRA)”. Cell which fails in the bucket1/2 test will be tagged and might be dropped according to DM1/2. Tagging is done by setting PNC bit in the cell RxBD and also in the last BD of the frame. If a cell is dropped, all remaining cells of that frame are dropped, except the last cell. The last cell can be dropped only if it is also the first cell or if the first cell was dropped. When frame or part of it, is dropped, FDC (Frame Drop Counter) is updated. If AAL5 frame is partially dropped, CRC error and length error will occur in the last RxBD of the frame.

30.2.20.2.3 GFR UPC

When UPC Table[UPCM]=11, the UPC works in GFR mode. In this mode, the following test are done on each cell:

1. Buckets test: **GCRA/F-GCRA**. Bucket1 and bucket2 are configured by UPC Table[LBF1] and UPC Table[LBF2]. A cell which fails in the bucket1/2 test will be tagged and might be dropped according to UPC Table[DM1/2]. Tagging is done by setting PNC in the cell BD and also in the last BD of the frame.

2. Maximum frame size (MFS). If the cell is not the last cell, the number of cells in frame until and include this cell, should be less than MFS. Cell which fails in this test will be dropped. If a cell is dropped, the PNC bit in the last BD of frame will be set.
3. CLP. The cell CLP should be the same as in the first cell. If it is not, the PNC bit in the last BD of the frame is set. If CLPDM=1 (CLP Drop Mode), the cell is dropped.

NOTE

There is no distinction between CLP0 to CLP1 change and CLP1 to CLP0 change.

If a cell is dropped, due to one of the three tests, all remaining cells of that frame are dropped, except the last cell. The last cell will be dropped only if it is also the first cell or if the first cell was dropped. When frame or part of it, is dropped, FDC (Frame Drop Counter) is updated. If frame is partially dropped, CRC error and length error will occur in the last RxBD of the frame.

GFR Conformance

The UPC supports GFR conformance testing according to the ATM Forum TM4.1 section 4.5.5.1. A frame is conforming if all its cells are conforming. A cell is conforming if it meets the following conditions:

1. The cell conforms to GCRA(1/PCR, CDVT) were PCR is defined for the CLP0+1 cell stream.
2. Its CLP value is identical to the CLP of the first cell.
3. The cell either is the last cell of the frame or the number of cells in frame up to and including this cell is less than MFS (Maximum Frame Size).

The configuration for GFR conforming:

1. UPC Table(UPCM)=11 -> GFR mode.
2. Leaky bucket 2 (Leaky bucket 1 is disabled at this point, and could be used later for F-GCRA) is configured to do GCRA(1/PCR,CDVT) for CLP0+1 cells stream, and drop any non-conformed cell.
3. UPC Table (CLPDM) set to 1, meaning that change in CLP will cause frame discard (if the user does not want to drop cells because of CLP change, CLPDM should be clear).
4. UPC Table (UPCMFS) should be assigned to the appropriate value.

GFR Frame-Eligibility (F-GCRA)

A frame is said to be “eligible” if and only if it meet the following conditions (ATM Forum TM 4.1 section 4.5.5.2):

1. It is conformed (see GFR conformance).
2. It passes the F-GCRA test.

In F-GCRA test only the first cell of the frame is tested. It should meets the following conditions:

1. CLP=0.
2. GCRA(1/MCR,BT+CDVT).

If the first cell meets those condition, the frame is considered as passing the test. The F-GCRA has two versions which differ in the TAT update. In ATM Forum TM specifications Version 4.1 annexB.5, if the frame passes the F-GCRA test or it is not conformed, the TAT is updated upon each cell arrival. In ATM Forum TM specifications Version 4.1 Appendix VI.2 if the frame pass the F-GCRA test, the TAT is updated upon each cell arrival. The QUICC Engine block UPC implements the F-GCRA Appendix VI.2 version.

The UPC configuration for eligibility test:

1. Setting GFR conforming test configuration, as in previous section.
2. Configure leaky bucket 1 to do the F-GCRA(1/MCR,BT+CDVT) test.
3. Configure the UPC Table[DM] bit: if set then frame which fails the F-GCRA test will be dropped, and if cleared frame which fails the F-GCRA test will be tagged by setting PNC bit in its RxBD.

30.2.20.3 Examples of Dual Leaky Bucket Configurations

Some Leaky Bucket examples are shown in Table 30-11. The examples show Leaky Bucket programming for ATM Forum TM Specification 4.1 service Categories and ITU-T I.371 ATM Transfer Capabilities.

Table 30-11. Example to Dual Leaky Bucket Configurations

ATM Forum TM4.1 Service Categories	ITU-T I.371 ATM Transfer Capability	LBF1	LBF2
CBR.1	DBR Configuration 1	011	000
VBR.1	SBR Configuration 1	011	011
VBR.2	SBR Configuration 2	011	001
VBR.3	SBR Configuration 3	011	001
GFR.1	—	100	011
GFR.2	—	100	011
UBR.1	—	011	000
UBR.2	—	011	000

30.2.20.4 UPC Statistics

The UPC manages a set of 16-bit counters for statistics that reside in the UPC table. When one of these counters overflows, a maskable interrupt can be generated to the host:

- CLP0 and CLP1 count the incoming cells.
- Nonconforming CLP0 counter (NCCLP0) and nonconforming CLP1 counter (NCCLP1) count the cells not conformed by the UPC buckets. A failure in one of the buckets causes the cell to be counted as nonconforming. NCCLP0CLP1/NCCLP1 applies to cells that arrived with CLP1.
- FC counter counts the incoming frames that were not dropped, meaning frames that were not counted by the FDC. This counter is applied only in frame mode (UPC Table[UPCM] = 10/11).
- FDC is the frame drop counter. When the UPC drops a frame or part of a frame, the FDC is updated. This counter is applied only in frame mode (UPC Table[UPCM] = 10/11).

- CDC is the cell drop counter. When the UPC drops a cell, the CDC is updated. This counter is applied only in cell base mode (UPC Table[UPCM] = 01).

30.2.21 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.
- miss-inserted dropped cell count—Counts cells discarded due to address look-up failure.

Counters are implemented in the multi-user RAM for each PHY device. The counters of each PHY are located in the UNI statistics table, described in Section 30.3.11, "UNI Statistics Table."

30.3 ATM Controller—Memory Map

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, the UNI statistics table and the UPC table.

30.3.1 Parameter RAM Page Organization

Figure 30-18 describes the organization of the parameter page for ATM mode. This structure of the page applies to the Distributor page as well as to the Thread page. The difference is the values programmed into different parameters page at initialization of the pages by the host. Each page is partitioned to sub pages starting at sub-page 0. Each sub-page contains parameters supporting different functionality. All the sub-pages have the same format which is depicted in the drawing. In addition to the sub-pages there is a single parameter - Local Page Parameter ptr which is a pointer to a table containing parameters associated with the current page. Detailed description will follow. Each of the sub-pages contains three fields which are actually pointers to data structures needed for supporting the functionality. First pointer is a pointer to a Configuration table which consists of all the fixed values parameters required by specific adaptation layers. The other two pointers are designated as a temporary storage used by the QUICC Engine block when processing a cell. On the distributor page one pointer is assigned for receiver temporary variables and the other is assigned for transmitter variables. In a thread page this two values are programmed to an identical values, which can act as receiver or transmitter variables depending on the thread functionality.

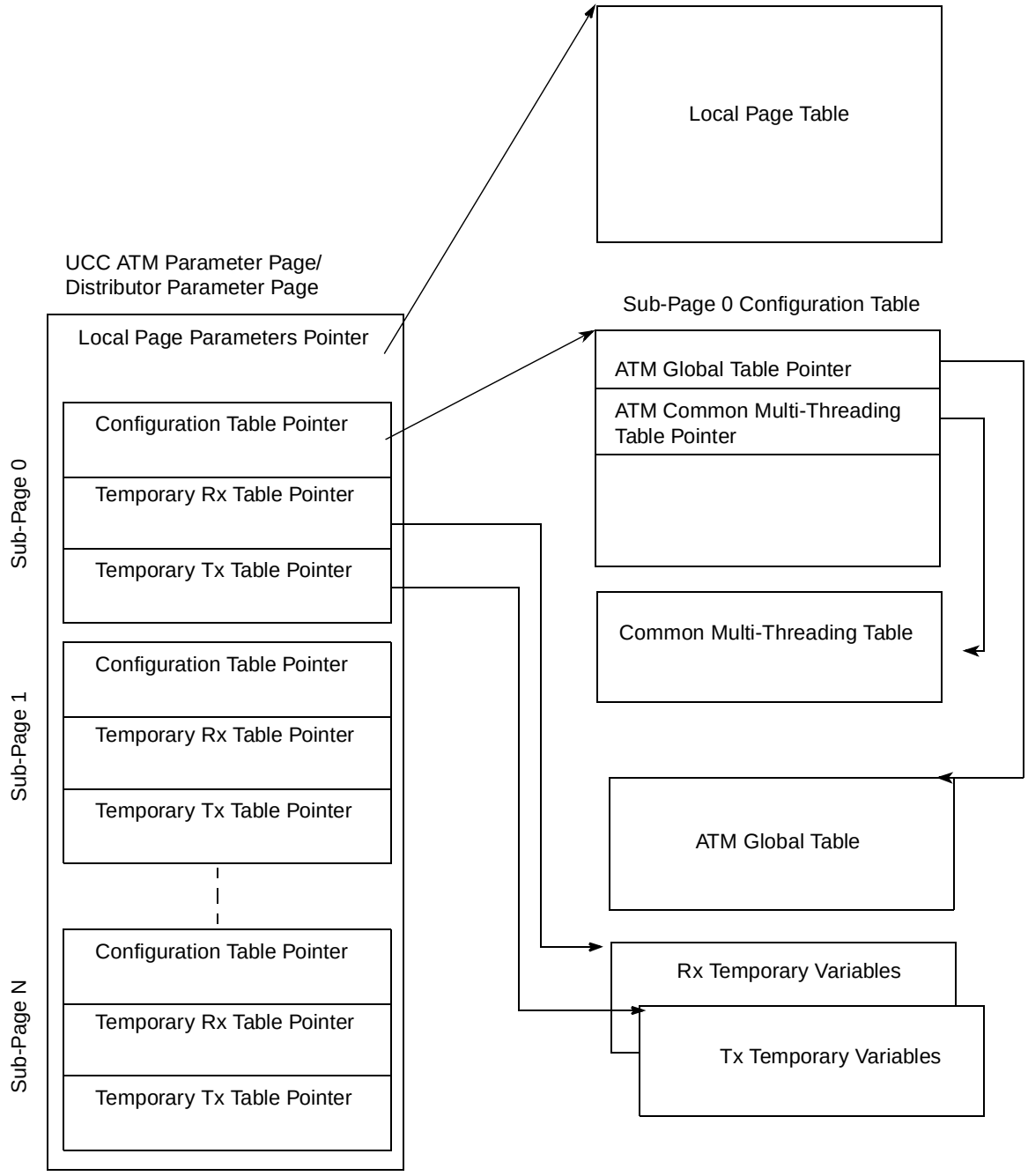


Figure 30-18. Parameter Page Organization

30.3.2 Parameter RAM

When configured for ATM mode, the UCC parameter page is mapped as shown in Table 30-12. When working in Non Multi-threading mode in the ATM level the UCC page is the only page used. When working in Multi-Thread mode the UCC page is called Distributor page. In this mode the Distributor page and the threads pages have the same structure. The host should initialize the distributor page with all the information required for operation of the UCC to which it is associated. The threads pages have to be

partially initialized while some of the parameters are assigned to the thread page dynamically during runtime.

30.3.2.1 Parameter RAM for Non Multi-Threading Operation

Table 30-12 describes the UCC page under Non Multi-Threading operation for the ATM level. There is a difference in memory allocation required for each of the modes. For details on parameter RAM on Multi-Threading mode see Section 30.3.2.2, "Parameter RAM for Multi-threading Operation."

Table 30-12. UCC Parameter RAM Page

Offset	Size	Field	Description	Memory Allocation
0x00	HW	**Local_page_ parameters_ptr**	Pointer to the table in the MURAM which contains parameters which are relevant and local to the current page. The address should be 8 bytes aligned.	Allocate 0x10 bytes. For content of the local parameter table see Table 30-13. **For PQ2 like operation, with 0x100 bytes page, should point to address of the UCC page+0x10.**
0x02	HW	**sub-page0_ Configuration_ Table_ptr**	This is a pointer to static configuration parameters table. Support for AAL0, AAL5. The address is 8 bytes aligned.	Allocate 0x60 bytes. see Table 30-18 for table content.The size of this table is 0x80 but last 0x20 bytes are for multi-threading operation and are not used. **For PQ2 like operation, with 0x100 bytes page, should point to address of the UCC page+0x20.**
0x04	HW	**sub-page0_Rx_ Tmp_Table_ptr**	Pointer to a table containing temporary Rx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40bytes. All table entries should be cleared. **For PQ2 like operation, with 0x100 bytes page, should point to address of the UCC page+0x80.**
0x06	HW	**sub-page0_Tx_ Tmp_Table_ptr**	Pointer to a table containing temporary Tx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40 bytes. Al entries in the table should be cleared. **For PQ2 like operation, with 0x100 bytes page, should point to address of the UCC page+0xC0.**
0x08	HW	**sub-page1_ Configuration_ Table_ptr**	**Initialize when Policer is enabled.** This is a pointer to static configuration parameters table. Address is 8 bytes aligned.	Allocate 0x40 bytes. see Table for table content.
0x0A	HW	**sub-page1_Rx_ Tmp_Table_ptr**	**Initialize when Policer is enabled.** Pointer to a table containing temporary Rx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40 bytes. All entries in table should be cleared.
0x0C	HW	**sub-page1_Tx_ Tmp_Table_ptr**	**Initialize when Policer is enabled.** Pointer to a table containing temporary Tx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40 bytes All entries in table should be cleared.

Figure 30-19 describes a possible arrangement for UCC Parameter Page which is 0x100 bytes. In this configuration all the sub-page0 parameters are contained in the thread page while sub-page1 parameters are located in a different location in the MURAM pointed by the Sub_page1_Rx/Tx_Tmp_Ptr's.

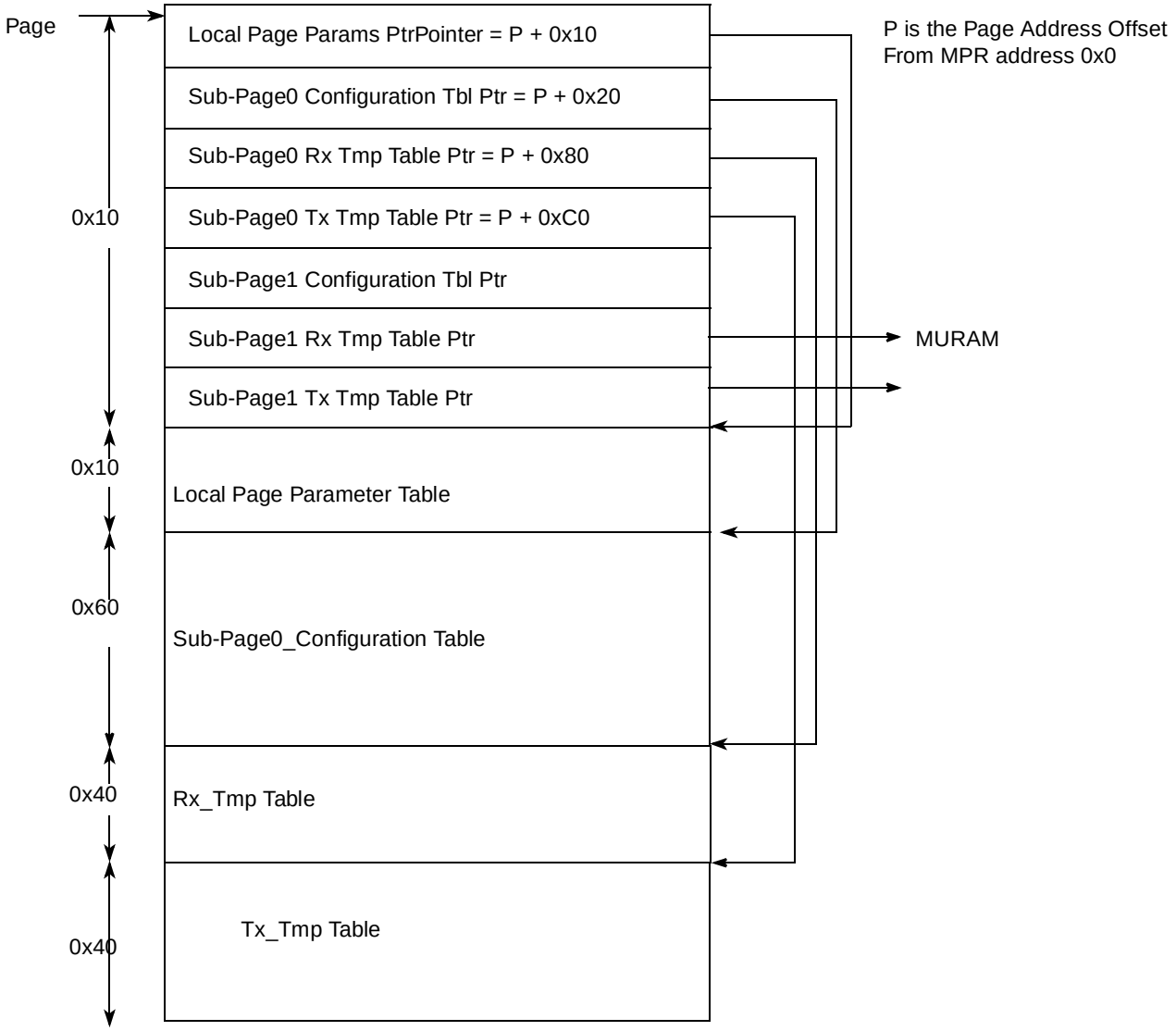


Figure 30-19. Possible Configuration for 0x100 Bytes UCC Page

Table 30-13 is pointed from distributor page see Table 30-12 [Local_page_parameters_ptr]. For Non-Multi-threading mode the size of this table is reduced to 0x10 bytes.

Table 30-13. UCC Local Page Parameter Table

Offset	Size	Field	Description	Non Multi-Threading Memory Allocation
0x00	HW	**General_Purpose TMP ptr**	Useful to fetch external RCT and AAL5 frame based external WFQ TCT.	Allocate 0x40 bytes. 32 bytes aligned.
0x02	HW	**IMA_Temp**	Memory location where an the IMA cell is built.	If IMA is enabled Allocate 0x80 bytes. 0x80 bytes aligned.
0x04	HW	**Rx TMP**	Memory location for receiver address look-up information.	Allocate 0x20 bytes.
0x06	HW	—	Reserved- Should be cleared	
0x08	HW	—	Reserved- Should be cleared	
0x0A	HW	—	Reserved- Should be cleared	
0x0C	W	—	Reserved- Should be cleared	

30.3.2.2 Parameter RAM for Multi-threading Operation

Table 30-14 describes the UCC page under Multi-Threading operation. In this mode the UCC page is called Distributor page. There is a difference in memory allocation required for each of the modes. For details on parameter RAM on Non Multi-Threading mode see Section 30.3.2.1, “Parameter RAM for Non Multi-Threading Operation.”

Table 30-14. Distributor Parameter RAM Page

Offset	Size	Field	Description	Multi-Threading Memory Allocation
0x00	HW	**Local_page_ parameters_ptr**	Pointer to the table in the MURAM which contains parameters which are relevant and local to the current page. The address should be 8 bytes aligned.	Allocate 0x10 bytes For content of the local parameter table see Table 30-15
0x02	HW	**sub-page0_ Configuration_ Table_ptr**	This is a pointer to static configuration parameters table. Support for AAL0, AAL5,. The address is 8 bytes aligned.	Allocate 0x80 bytes. See Table 30-18 for table content
0x04	HW	**sub-page0_Rx_ Tmp_Table_ptr**	Pointer to a table containing temporary Rx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40bytes. All table entries should be cleared.
0x06	HW	**sub-page0_Tx_ Tmp_Table_ptr**	Pointer to a table containing temporary Tx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40 bytes. Al entries in the table should be cleared.

Table 30-14. Distributor Parameter RAM Page (continued)

Offset	Size	Field	Description	Multi-Threading Memory Allocation
0x08	HW	**sub-page1_ Configuration_ Table_ptr**	**Initialize when Policer is enabled.** This is a pointer to static configuration parameters table. Address is 8 bytes aligned.	Allocate 0x40 bytes. see Table for table content.
0x0a	HW	**Reserved**	Reserved. Should be cleared.	No Allocation needed
0x0c	HW	**Reserved**	Reserved. Should be cleared.	No Allocation needed

Table 30-15 is pointed from distributor page see Table 30-14 [Local_page_parameters_ptr]. For Multi-Threading mode the table has to be fully initialized.

Table 30-15. Distributor Local Page Parameter Table

Offset	Size	Field	Description	Multi-Threading Memory Allocation
0x00	HW	**Reserved**	Reserved. Should be cleared.	No allocation is needed
0x02	HW	**Reserved**	Reserved. Should be cleared.	No allocation needed
0x04	HW	**Rx TMP**	Memory location for receiver address look-up information in multi-threading mode.	Allocate 0x20 bytes.
0x06	HW	**Reserved**	Reserved. Should be cleared.	No allocation needed
0x08–0x10	24 bytes	**Reserved**	Reserved. Should be cleared.	—

Table 30-16 describes the Thread parameter table which has to be assigned and initialized for each of the ATM threads available in the system.

Table 30-16. Thread Parameter RAM Page

Offset	Size	Field	Description	Memory Allocation
0x00	HW	**Local_page_ parameters_ ptr**	Pointer to the table in the MURAM which contains parameters which are relevant and local to the current page. The address should be 8 bytes aligned.	Allocate 0x30 bytes. See Table 30-17 for initialization details.
0x02	HW	**sub-page0_ Configuration _Table_ptr**	This is a pointer to static configuration parameters table. Support for AAL0, AAL5. The address is 8 bytes aligned.	This entry is passes to a thread page by the Distributor Dynamically. **Initialized to 0.**
0x04	HW	**sub-page0_Rx _Tmp_Table_ ptr**	Pointer to a table containing temporary Rx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40 bytes. Should be equal to the pointer in: sub-page0_Tx_Tmp_Table_ptr.
0x06	HW	**sub-page0_Tx _Tmp_Table_ ptr**	Pointer to a table containing temporary Tx variables and parameters. Address is 8 bytes aligned.	The same 0x40 bytes allocated in sub-page0_Rx_Tmp_Table_ptr.

Table 30-16. Thread Parameter RAM Page (continued)

Offset	Size	Field	Description	Memory Allocation
0x08	HW	**sub-page1_ Configuration _Table_ptr**	This is a pointer to static configuration parameters table. Address is 8 bytes aligned.	This entry is passes to a thread page by the Distributor Dynamically. **Initialized to 0.**
0x0a	HW	**sub-page1_Rx _Tmp_Table_ ptr**	Pointer to a table containing temporary Rx variables and parameters. Address is 8 bytes aligned.	Allocate 0x40 bytes. Should be equal to the sub-page1_Tx_Tmp_Table_ptr.
0x0c	HW	**sub-page1_Tx _Tmp_Table_ ptr**	Pointer to a table containing temporary Tx variables and parameters. Address is 8 bytes aligned.	The same 0x40 bytes allocated in sub-page1_Rx_Tmp_Table_ptr.

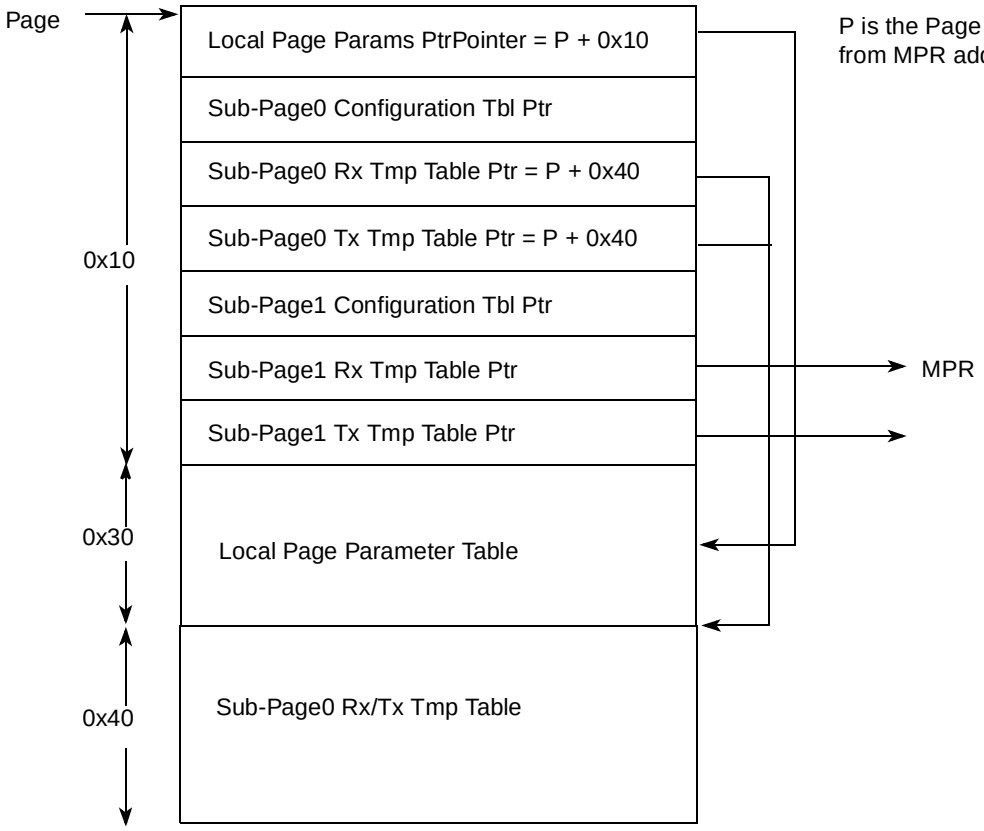


Figure 30-20. Possible Configuration for a Thread Page

Figure 30-20 describes a possible arrangement for a Thread Parameter Page which is 0x80 bytes. In this configuration all the sub-page0 parameters are contained in the thread page. There is no need for the Sub-Page Configuration table in the thread page. Sub-page1 Rx/Tx Tmp tables parameters are located in a different location in the MURAM pointed by the Sub_page1_Rx/Tx_Tmp_Ptr's.

Table 30-17. Thread Local Page Parameter Table

Offset	Size	Field	Description	Memory Allocation
0x00	HW	**General_Purpose TMP ptr**	Useful to fetch external RCT and AAL5 frame based external WFQ TCT.	Allocate 0x20 bytes. 32 bytes aligned
0x02	HW	—	reserved should be cleared.	—
0x04	HW	—	reserved should be cleared.	—
0x06	HW	—	reserved should be cleared.	—
0x08	HW	—	reserved should be cleared.	—
0x0A–0x2F	—	—	Reserved. Should be cleared.	—

Table 30-18 contains all the fixed value parameters configured by the host for the ATM operation. This table is pointed from the distributor page and is passed to each thread dynamically upon assignment to a distributor. This sub-page contains parameters which are general to the ATM and IMA and parameters for AAL0, AAL5. This table is pointed from distributor page Table 30-12 [sub-page0_Configuration_Table_ptr].

Table 30-18. Sub-page0 Configuration Table

Offset	Size	Field	Description	Memory Allocation
0x00	HW	**Global ATM Parameters Table ptr**	A pointer to a Common ATM parameters for all the UCCs see Table 30-19 for table content	Allocate 0x30 bytes. 32 bytes aligned
0x02	HW	**OAM CH RCT PTR**	A pointer to OAM RCT. This RCT is a designated channel for RAW cell queue On PQII family this was channel #1 of each UCC	Allocate 0x20 bytes. 32 bytes aligned.
0x04	HW	**IMAROOT**	Pointer to the IMA root parameter table.	If IMA enabled Allocate 0x80 bytes. 128 byte aligned
0x06	HW	**UCC_Modes**	Defines mode of operation for this UCC. See 30.3.2.6/30-58.	—
0x08	W	**Reserved**	Reserved. Should be cleared.	—
0x0c	HW	**GMODE**	Global mode. User-defined. See 30.3.2.5/30-57.	—
0x0e	HW	**INT_TCTE_TMP_ptr**	A temporary area for fetching the external TCTE	Allocate 0x20 bytes.
0x10	HW	**ADD_COMP_LOOKUP_BASE**	This is a pointer to the Address compression lookup table described inTable 30-26.	points to a 12 bytes table (if address compression is enabled).

Table 30-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Memory Allocation
0x12	HW	**DYN_ADD_COMP_BASE /Mini-CAM Look-up table base**	If using the dynamic change of address compression table this is the pointer to the alternate table If using Mini CAM address look-up this entry points to the parameter table for this mode described in Table 30-27.	points to a 12 bytes table.
0x14	HW	**Receiver- Time-out request period**	This value determines a time-out period for request asserted to the QUICC Engine ATM receiver. It is measured in UTOPIA clocks count. If **UCC_Modes[NPL]** is set **or GMODE[ALM] =1x** program this value to zero	NA
0x16	HW	**VCI_FILTER**	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 30.3.2.4/30-56.	—
0x18	HW	**APCP_BASE**	APC parameter table base address. User-defined offset from multi-user RAM base.	Number of PHY's *32 bytes
0x1A	HW	**FBT_BASE**	Free buffer pool parameter table base. User-defined offset from multi-user RAM base.	Number of Buffer pools *32 bytes (Up to 4 pools)
0x1C	HW	**INTT_BASE**	Interrupt queue parameter table base. User-defined offset from multi-user RAM base.	Number of queues *32 (Up to 4 queues)
0x1E	HW	**UNI_STATT_BASE**	UNI statistics table base. User-defined offset from multi-user RAM base. Note that this must be set up according to 30.3.11/30-117.	Number of PHY's * 4 bytes
0x20	HW	**UEAD_OFFSET/ CHANNEL_CODE_OFFSET**	User-defined cells mode only. Offset to the user-defined extended address (UEAD) in the UDC extra header. When GMODE[ALM]=1x this entry points to the offset in the UDC where the CC resides. It Must be an even address. See 30.3.2.3/30-56.	—

Table 30-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Memory Allocation
0x22	HW	**IDLE_BASE**	Valid only when Serial ATM is enabled or when “external rate mode” is enabled. Idle/unassign cell base address. Points to multi-user RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined offset from multi-user RAM base. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP=1).	—
0x24	HW	**IDLE_SIZE**	Valid only when Serial ATM is enabled or when “external rate mode” is enabled. Idle/unassign cell size. 52 in regular mode; 68 in case of UDC of size 1-8 bytes, 76 in case of UDC of size 9-12 bytes.	—
0x26	Byte	**BD_BASE_EXT (MSB)**	The MSB byte of the base address for the BDs.	—
0x27	Byte	—	Reserved Should be cleared	—
0x28	W	—	Reserved Should be cleared	—
0x2C	HW	—	Reserved Should be cleared	—
0x2E	HW	**PMT_BASE**	Performance monitoring table base. User-defined offset from multi-user RAM base.	Number of Tables * 32 (Up to 64 tables)
0x30	W	—	Reserved Should be cleared	—
0x34	HW	—	Reserved Should be cleared	—
0x36	HW	—	Reserved Should be cleared	—
0x38	W	—	Reserved Should be cleared	—
0x3c	HW	—	Reserved Should be cleared	—
0x3e	HW	—	Reserved Should be cleared	—
0x40	W	—	Reserved Should be cleared	— —
0x44	HW	—	Reserved Should be cleared	—
0x46	HW	—	Reserved Should be cleared	—
0x48	HW	—	Reserved Should be cleared	—
0x4a	HW	—	Reserved. Should be cleared	—
0x4c	HW	—	Reserved Should be cleared	—

Table 30-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Memory Allocation
0x4e	HW	—	Reserved Should be cleared	64 bytes
0x50	W	—	Reserved Should be cleared	—
0x54	W	—	Reserved Should be cleared	—
0x58	W	—	Reserved Should be cleared	—
0x5c	W	—	Reserved Should be cleared	—
0x60	Byte	**Rx Terminator SNUM**	The SNUM which is used for the receiver terminator process. For available thread numbers see Table 19-17. Needed only for ATM level 1 MTH mode.	—
0x61	Byte	**Tx Terminator SNUM**	The SNUM which is used for the transmitter terminator process. For available threads numbers see Table 19-17. Needed only for ATM level 1 MTH mode.	—
0x62	HW	**Common MTH parameters Table Base**	**Valid in multi-Threading mode.** Pointer to the Common Multi thread page described in Table 30-20.	Points to a 0x20 byte table.
0x64	HW	**MTH_Terminator_Rx_Status_ptr**	**Valid in multi-Threading mode only.** All 66 bytes Rx terminator status should be cleared.	Allocates 0x42 bytes. This pointer should be 0x80 bytes aligned.
0x66	HW	**MTH_Terminator_Tx_Status_ptr**	**Valid in multi-Threading mode only.** All 66 bytes Tx terminator status should be cleared.	Allocates 0x42 bytes. This pointer should be 0x80 bytes aligned.
0x68	W	**ATM_Available_Rx_Thread_ MASK**	32 bit indicating available threads for this receiver UCC. See Figure 30-21.	— —
0x6C	W	**ATM_Available_Tx_Thread_ MASK**	32 bit indicating available threads for this transmitter UCC. See Figure 30-21.	—

Note: The value of the time-out should be determined based on the application of the ATM traffic for this UCC. In order to optimize performance of the QUICC Engine block, the ATM traffic received is processed in pipe-line where each cell triggers completion of processing a previous cell. In cases where there is a period of idle on the line it potentially cause a stall in the pipe where last cell is not processed. The value of this time-out defines an allowed idle period on the UTOPIA bus for this UCC. If the timeout has expired, a request is asserted to the RISC in order to finish processing the last cell in the FIFO. The value determines a maximum latency allowed on the system in cases of a bursty traffic. A minimum value of this timer should be a cell time slot in the expected rtae.When setting a too low value this could cause over requesting of the QUICC Engine block thus causing no efficient work of the RISC. When setting a higher value it could cause a higher latency on a last cell in a burst. Setting this entry to zero disables this feature.

30.3.2.2.1 ATM_Available_Xx_Thread_Mask Description

Each bit in this register indicates availability of the relevant thread number to this distributor. Numbering of the threads is according to their location in the ATM Thread Table described in Section 30.2.3, "ATM Multi-Threading." Setting a bit indicates the thread is available and clearing the bit indicates the thread is not available. When a user wants to designate a thread uniquely to a specific UCC the bit corresponding to this thread is set exclusively for this distributor. A thread which is available for operation under several UCC will have its bit set in each of the distributors pages of these UCCs. The thread number is also referred as Thread ID and are assigned in a consecutive manner. At least two threads should be assigned per UCC. One for receive and one for transmit operation.

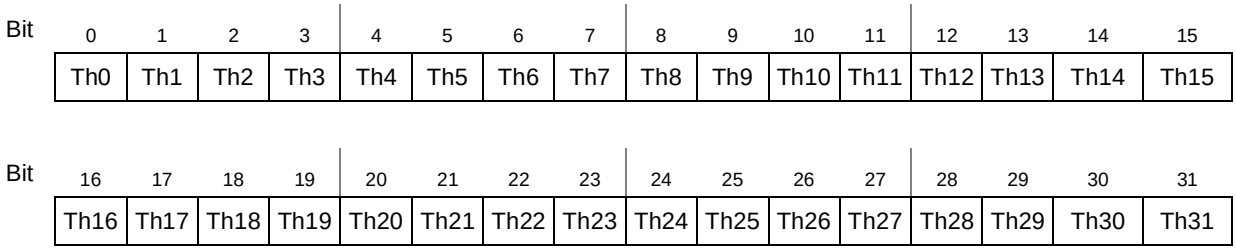


Figure 30-21. ATM_Available_Xx_Thread_Mask

Table 30-19 shows the Global ATM parameters Table. It is pointed from Table 30-18 [Global ATM Parameters Table ptr].

Table 30-19. Global ATM Parameters Table

Offset	Size	Field	Description
0x00	HW	**INT_RCT_BASE**	Internal receive connection table base. User-defined offset from multi-user RAM base. Should be common value for all the ATM UCCs.
0x02	HW	**INT_TCT_BASE**	Internal transmit connection table base. User-defined offset from multi-user RAM base. Should be common value for all the ATM UCCs.
0x04	W	**EXT_RCT_BASE**	External receive connection table base. User-defined. Should be common value for all the ATM UCCs.
0x08	W	**EXT_TCT_BASE**	External transmit connection table base. User-defined. Should be common value for all the ATM UCCs.
0x0C	W	**EXT_TCTE_BASE**	External transmit connection table extension base. User-defined. Should be common value for all the ATM UCCs.
0x10	HW	**INT_TCTE_BASE**	Internal transmit connection table extension base. User-defined offset from multi-user RAM base. Should be common value for all the ATM UCCs.
0x12	HW	**COMM_INFO_CTL**	The information field associated with the last host command. User-defined. See Section 30.4.1, "ATM Commands."
0x14	HW	**COMM_INFO_CC**	
0x16	HW	**COMM_INFO_BT**	
0x18	W	COMM_SUS_TMP	Command. Reserved. Should be initialized to 0.
0x1C	HW	COMM_LK_TMP	Command. Reserved. Should be initialized to 0.
0x1E	HW	—	Reserved. Should be cleared

Table 30-19. Global ATM Parameters Table (continued)

Offset	Size	Field	Description
0x20–0x30	16 Bytes	—	Reserved. Should be cleared.

Table 30-20 contains the Common MTH Parameters. It is pointed from Table 30-18 [Common MTH parameters Table Base]. The table contains all the management parameters for the Multi-threaded operation for the ATM. Several UCCs should point to this table since the threads are a common resource and are shared between the UCC's. In general when UCC_MODES(ATM_lvl_MT_en) is set, this table has to be initialized. If ATM level operates in Non-Multi-threaded operation this table has to be assigned and initialized as instructed in the table. If only ATM level is required in multi-threaded operation only the first 10 bytes should be initialized (Allocation is still for 0x20 bytes).

Table 30-20. Common MTH Parameters Table

Offset	Size	Field	Description	Memory Allocation
0x00	HW	**ATM_Threads_Table_Base**	Points to a table in MURAM which contains the list of all available threads for this group as described in Table 30-30 ATM thread table.	Number of threads* 4 bytes.
0x02	HW	**ATM_Thread_CAM_BASE**	Offset to a table in MURAM which has 4 bytes per entry. Each entry in this table should be initialized to 0.	Number of threads* 4 bytes
0x04	W	**ATM_Thread_Empty_Status**	Thread Empty Status. This structure consist 1 bit per thread, which represents if the thread is empty or not. On initialization, the number of zeros in this parameter is equal to the number of threads. For example if there are 10 ATM threads, then initialize 0x003F-FFFF.	—
0x08	Byte	**IW**	The size of the (ATM_thread_CAM table in bytes-1). equal to (Number of threads* 4 -1).	—
0x09–20	—	—	—	—

Table 30-21 consists all the fixed value parameters configured by the host for the additional functionality of the ATM operation. This table is pointed from the distributor page and is passed to each thread dynamically upon assignment to a distributor. This sub-page contains parameters which are needed for policer operation.This table is pointed from Table 30-12 [sub-page1_Configuration_Table_ptr].

Table 30-21. Sub-page1 Configuration Table

Offset	Size	Field	Description
0x00	W	—	Reserved. Should be cleared

Table 30-21. Sub-page1 Configuration Table (continued)

Offset	Size	Field	Description
0x04	W	—	Reserved. Should be cleared
0x08	HW	—	Reserved. Should be cleared
0x0A	HW	—	Reserved. Should be cleared.
0x0C	W	—	Reserved. Should be cleared
0x10	W	**EXT_UPC_BASE**	Standalone UPC Mode: External UPC Table base. User-defined. Should be aligned to 64 byte address.
0x14	HW	**INT_UPC_BASE**	Standalone UPC Mode: Internal UPC Table base. User-defined. Should be aligned to 16 byte address.
0x32–0x3F	—	—	Reserved. Should be cleared.

30.3.2.3 Determining UEAD_OFFSET (UEAD Mode Only)

The UEAD_OFFSET value described in Table 30-18 is based on the position of the user-defined extended address (UEAD) in the UDC extra header. Table 30-22 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

Table 30-22. UEAD_OFFSETs for Extended Addresses in the UDC Extra Header

30.3.2.4 VCI Filtering (VCIF)

VCI filtering enable bits are shown in Figure 30-22.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—			VC3	VC4	—	VC6	VC7	VC8	VC9	VC10	VC11	VC12	VC13	VC14	VC15

Figure 30-22. VCI Filtering Enable Bits

Table 30-23 describes the operation of the VCI filtering enable bits.

Table 30-23. 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.

30.3.2.5 Global Mode Entry (GMODE)

Figure 30-23 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	—		GBL	—			ALB	CTB	REM	0	IMA_EN	UEAD	CUAB	EVPT	ALM	

Figure 30-23. Global Mode Entry (GMODE)

Table 30-24 describes GMODE fields.

Table 30-24. GMODE Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global. Asserting GBL enables snooping of external connection tables and address look-up tables. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
3–5	—	Reserved, should be cleared.
6	ALB	Address look up bus for CAM or address compression tables 0 Reside on the coherent system bus. 1 Reside on the QUICC Engine secondary bus.
7	CTB	External connection tables bus 0 Reside on the coherent system bus. 1 Reside on the CE secondary bus.
8	REM[1]	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	—	Reserved, should be cleared.
10	IMA_EN	Enable the associated UCC in IMA mode. 0 Default. UCC Enabled in normal ATM mode 1 UCC enabled in IMA mode Note that individual PHYs of those IMA-mode enabled UCCs may still be set for non-IMA functionality via the IMAPHY register in the IMA root table. **Important**: IMA mode is operable in IMA NON MULTI-threaded mode only so that UCC_Modes[ATM_lvl_MT_en] should be cleared.

Table 30-24. GMODE Field Descriptions (continued)

Bits	Name	Description
11	UEAD	User-defined cells extended address mode. See 30.2.16.1/30-30. 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 multi-user RAM. 1 VP table reside in external memory.
14–15	ALM	Address look-up mechanism. See 30.2.14/30-22. 00 Reserved. Should not be used. 01 All PHYs Address compression mode. 10 All PHYs Channel code is in UDH (extra header), location programmed in CH_CODE_OFFSET. No address look-up is performed and no Protection mode. 11 All PHYs Channel code is in UDH. Protection Mode is enabled. Channel code appears twice in UDH, and cell is discarded if values are not identical. CH_CODE_OFFSET points to the first occurrence of the channel code. **Note:** In Case Ucc_Modes[MCAL] is set the Mini-Cam lookup is overriding this setting unless there is a PHY that does need one of these modes

Note:

1 GMODE[REM] must be set to disable receive emergency mode. If receive emergency mode were enabled, it would result in IMA transmit protocol violations in cases of system overload, potentially avoiding protocol errors in the IMA receiver at the expense of generating IMA transmit protocol violations to the far end. Rather than causing erratic transmit operation, it is better to set GMODE[REM] and deal with the IMA receive protocol violations locally. Note further that the system should be designed such that overload problems never occur in the field; any errors due to overload should be eliminated during system design and debug.

NOTE

For better system performance and better bus utilization it is recommended to have the Look-up tables and the connection tables to reside on a different bus than the bus used for the data buffers or BD's. In this way the load of the buses and the DMA latency when accessing these data structures is decreased and performance increase.

30.3.2.6 UCC_Modes

UCC_Modes entry is a 16 bit register used to define mode of operation for the UCC. Each bit enables a certain feature and some could be changed dynamically and used as a trigger for QUICC Engine block operation. Features are described in Table 30-25.

	0	1	2	3	4	5	6	7	8	9	10	11	15
Field	ATM_1v1_MT_en	IDLE_ModEn	TSR_Sel	—		DYN_ACEn	UID		VPSW_En	NPL	MCAL	—	

Figure 30-24. UCC Modes Entry

Table 30-25. UCC_Modes Field Descriptions

Bits	Name	Description
0	**ATM_Ivl_MT_en**	This bit specified operation in Multi-threaded mode for the ATM level. 0 No Multi-threading mode 1 Multi-thread operation is enabled. All the parameters required should be initialized and a special host command should be issued. **Important**: when IMA mode is enabled this bit should be cleared.
1	IDLE_Mod En	0 No Idle cells are transmitted. This is for normal operation of the ATM where the ATM traffic is on a Utopia bus using “internal rate mode”. 1 Enable transmission of idle cells. This is when Serial ATM code is performing the TC layer or when “external rate mode” is enabled and Idle cells are required.
2	**TSR_Sel**	QUICC Engine timer prescale select for transmitter scheduling for example, GCRA scheduler and APC flux mechanism. 0 CETSCR1 register is used to determine the time units using CETPS1. 1 CETSCR2 register is used to determine the time units using CETPS2. **Note:** For other timing involved parameters CETSCR1 is always used.
3–4	**—**	Reserved. Should be cleared
5	**DYN_AC En**	Dynamic address compression mode Enable. Set by the core when a dynamic change of address compression tables is needed. Reset by the QUICC Engine block upon completion. If set by the host it indicates the QUICC Engine block to swap sub-Page0 Configuration table ADD_COMP_LOOKUP_BASE parameter with DYN_ADD_COMP_BASE parameter and the new table is being used for the address compression.
6–7	UID	UTOPIA Bus ID. These two bits are assigned at Init by the RISC and are used for address compression and for the APC table’s location. The assignment is the following: UCC1, UCC3, UCC5 have the UID=0b00, and UCC2, UCC4 have the UID=0b01. The user should not change this value after issuing Init host command.
8	**VPSW_En**	VP Switch mode enable. 0 PQII like operation. When GMODE[CUAB] is set, Upon detecting unallocated bits in the VC level the QUICC Engine block will discard the cell. 1 The QUICC Engine block will access offset 0 in the selected VCLT and look for a valid channel to be used as the destination channel for this VP cell. In case the MS bit in this entry is set the cell is dropped.
9	**NPL**	Non Pipe-lined. 0 The QUICC Engine block utilizes a pipe-line in the address look-up process. This bit is cleared when the UCC is connected to the UTOPIA interface and no IMA or serial ATM is enabled or when mini-CAM address look-up is disabled on this UCC. 1 The QUICC Engine block does not utilize a pipe-line in the address look-up process. This bit is set if this UCC is working with IMA enabled or when working under Serial ATM or when working in mini-CAM look-up
10	**MCAL**	Mini-Cam Address Look-up Mode. 0 The address look-up is based on GMODE[ALM] 1 The channel code is determined by an internal mini CAM table on a per PHY as described in 30.2.14.2, “Mini-CAM Address Look-Up
11–15	—	Reserved, should be cleared.

NOTE

The structure of the VP level is designed for future expansion. The PHY ID field contains 8 bits, supporting up to 256 PHYs per UTOPIA interface. The UID field contains two bits supporting up to four UTOPIA interfaces. PHY ID should be treated as the following: 5 LSBs represent the 5 address lines of the UTOPIA interface. Three MSBs represent the device ID. For this reason, the application should clear the bits which are not applicable for the device. Table 30-26 shows the setting for the VP level mask field in the Address Look-up table. This table contains the 4 bytes from offset 0x8 in the address look-up table.

30.3.2.7 Address Look-Up Table

The UCC is able to support up to 127 PHYs addresses. Eight address bits are available for usage as baseline for the address compression look-up. In cases the address compression tables are used globally for the QUICC Engine block, meaning all UCCs use the same address compression tables it is needed to differentiate identical PHY,VPI,VCI received on different utopia buses by their utopia bus ID. The ID of each utopia bus is assigned automatically by the QUICC Engine block at initialization. It has 2 bits assigned for this. The total number of bits entering the VP level address compression is up to 22 bits where 12 bits are for the VPI of the cell, 8 bits are for up to 256 PHYs ID's and 2 more bits indicating the utopia bus ID. Table 30-26 contains the Address Look-Up parameters, It consists of three parameters as described in the table. It is pointed by Table 30-18 sub-page0 configuration table [ADD_COMP_LOOKUP_BASE].

Table 30-26. Address Look-Up Table

Offset	Name	Width	Description
0x00	VPT_BASE	Word	Base address of the address compression VP table. User-defined. If GMODE[EVPT]= 0 only 2 bytes offset from MURAM should be initialized.
0x04	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x08	VCLT_SF	1 Byte	VC level table Scale factor 0x00- default. The address to the VC level table is calculated by VCT_BASE + VCOFFSET *4 0x01- a scale of 2 is added. Address is VCT_BASE + VCOFFSET *8 0x02- a scale of 4 is added. Address is VCT_BASE + VCOFFSET *16 0x03- a scale of 8 is added. Address is VCT_BASE + VCOFFSET *32 0x04- a scale of 16 is added. Address is VCT_BASE + VCOFFSET *64 If a larger scale factor is needed, this field could be programmed accordingly, for example, 0x05 for a scale factor of 32, etc.
0x09	VP_LVL_MASK	3 bytes	A mask set by the s/w which determines which of the 22 available address information bits are used in the address compression operation as described in Section 30.2.14.1, "Address Compression." See Figure 30-25 description of how to program the mask bits.

Note: The structure of the VP level is designed for future expansion. The PHY ID field contains 8 bits supporting up to 256 PHYs per UTOPIA interface and the UID field contains 2 bit supporting up to four Utopia i/f. For this reason the application should clear the bits which are not applicable for the device. Table 30-25 shows the setting for the VP Level mask field in the Address Look-up table.This table contains the 4 bytes from offset 0x8 in the address look-up table.

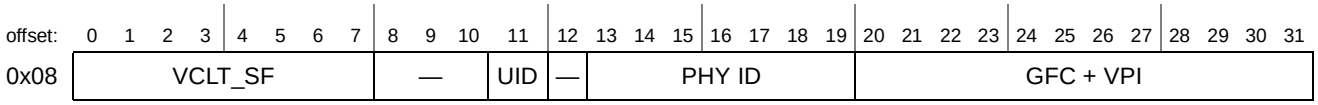


Figure 30-25. VP_LVL_MASK

30.3.2.8 Dynamic Address Compression Table change

The user may want to change dynamically the VPT_BASE and the VCT_BASE parameters, in order to support dynamic and more complex VP/VC ranges for the Address Compression mode.

This change is done automatically in synchronization with the ATM receiver flow. The Host should set the UCC_Modes[DYN_En] bit located in sub-page0 configuration table. As a reaction the QUICC Engine block will swap the ADD_COMP_LOOKUP_BASE with the DYN_ADD_COMP_BASE and from that point will use the alternative table as a base for the address compression calculations.

Upon completion of the dynamic change process the QUICC Engine block clears the DYN_AC_En entry. Until then the CPU shouldn't toggle the DYN_AC_En entry.

30.3.2.9 Dynamic Address Look-Up Table

This is an identical table to the one described in Table 30-26. This is an alternative table which the user can initialize and use in case of a dynamic address compression change.

30.3.2.10 Mini-CAM Address Look-up Table

This data structure replaces the Dynamic Address compression Table and contains the UCC parameters required for the Mini-CAM look-up operation described in Section 30.2.14.2, "Mini-CAM Address Look-Up.' For each incoming ATM cell the header is searched in the Mini-CAM tables for a match. Upon a match a channel code is calculated for this VPI/VCI.

Table 30-27. Mini-CAM Look-Up Table

Offset	Name	Width	Description
0x00	Base_Channel#	HW	This is the base channel number for calculations of the ATM channel code. The final channel number assigned to an incoming cell on a match is determined using: Channel# = Base_Channel# + PHY_ID * Max_MC_size + Mini-CAM entry index
0x02	Mini-CAM_base	HW	Base offset in the MURAM to the mini CAM tables. Each entry is a four bytes entry described in Figure 30-27.
0x04	PHY_Table_base	HW	Base offset in the MURAM offset, to the PHY table. Each entry is a two bytes entry described in Figure 30-26.
0x06	Max_MC_size	byte	Number of entries in the largest mini CAM table in the system. Size of this table could be one of the following: 8,16,32,64. The value programmed in Max_MC_size is the power of 2 of the size so values are 3,4,5,6 respectively.
0x07	Max_UPC_Entries	byte	If Policing is enabled in this PHY up to 15 PID (Policer ID) are available for each PHY. This value is maximum value of UPCs used for any of the PHYs. This value should be rounded to the next power of 2. The value programmed in Max_UPC_Entries is the power of 2 of the size (for example if the maximum number of UPCs used per PHY is 5 set this value to 3)

Table 30-27. Mini-CAM Look-Up Table (continued)

Offset	Name	Width	Description
0x08	First_PID	HW	The first Policer ID used in the system. Any of the UPCs used is calculated based on this number. The final PID number assigned to an incoming cell is determined using: PID# = First_PID + PHY_ID * Max_UPC_Entries + mini-CAM entry[PID]

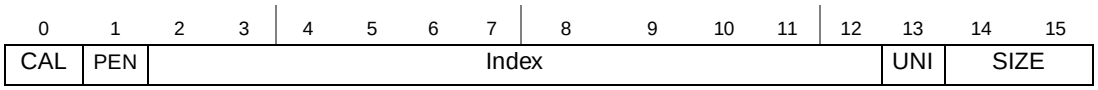


Figure 30-26. PHY Table Entry

Table 30-28. PHY Table Entry Description

Bit	Name	Description
0	CAL	Classical address compression look-up for this PHY. No mini-CAM operation
1	PEN	Policer Enable for this PHY
2–12	Index	Index to the mini-CAM table for this CAM. Address of the table is: Mini-CAM_base + Index*2^5
13	UNI	UNI mode. 0 Take the GFC value into the CAM search. 1 Mask the GFC field in the CAM search
14–15	Size	Size of mini-CAM table 00 size is 8 entries 01 size is 16 entries 10 size is 32 entries 11 size is 64 entries

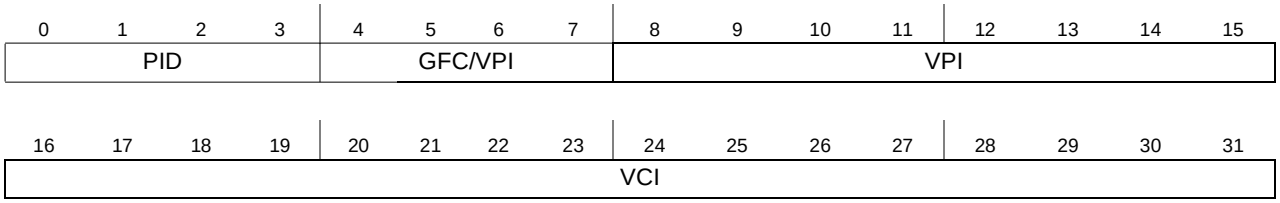


Figure 30-27. Mini-CAM Entry

Table 30-29. Mini-CAM Entry Description

Bit	Name	Description
0–3	PID	If PID=0x0 policer is disabled for this VPI/VCI If PID>0 Policing is enabled for this channel and the Policer ID is calculated by: PID# = First_PID + PHY_ID * Max_UPC_Entries + mini-CAM entry[PID]
4–7	GFC/VPI	The GFC/VPI value of the cell. If UNI bit is set in the PHY table entry this bits are ignored on the incoming cell.
8–15	VPI	VPI of the ATM cell

Table 30-29. Mini-CAM Entry Description (continued)

Bit	Name	Description
16–31	VCI	VCI of the ATM cell

30.3.3 Multi-Threading Structures

When working in Multi-Threading mode, see Section 30.2.3, “ATM Multi-Threading,” the user defines a pool of the available threads for processing the ATM traffic. The threads are available SNUM of the QUICC Engine block RISC and the user should initialize a table which contains a list of all the designated SNUMs. ThreadID number is according to its location in the table. First entry is thread 0, second entry is thread 1 etc. Figure 30-28 depicts an entry of the ATM Threads Table, which resides in the RAM and consists of as many entries as allocated by the user (up to 28 entries). The table is pointed from the Common MTH parameters Table[ATM_Threads_Table_Base].

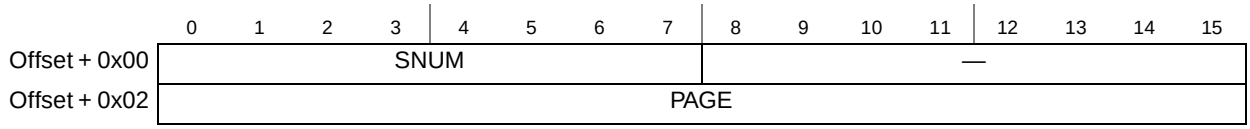


Figure 30-28. ATM Threads Table Entry

Table 30-30. ATM Threads Table Entry Description

Offset	Bits	Name	Description
0x00	0–7	SNUM	Back End SNUM. For available thread numbers see Table 19-17.
	8–15	—	Reserved. Should be cleared.
0x02	—	PAGE	Page pointer.

30.3.4 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 address compression tables or the mini-CAM tables and when the QUICC Engine block sends an interrupt to an interrupt queue.

Example:

Suppose a configuration supports 1,024 regular ATM channels. To allocate 4 Kbytes of multi-user 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 30-29.

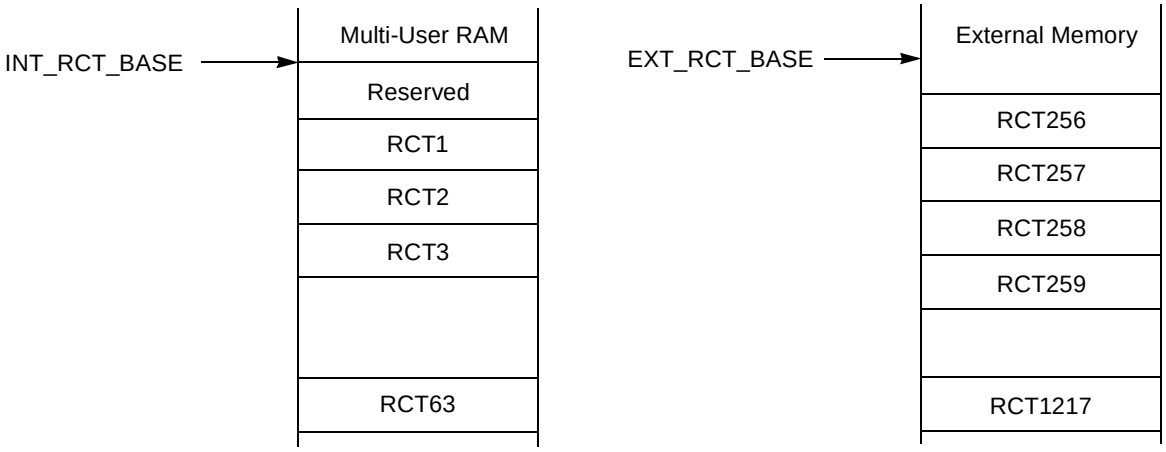


Figure 30-29. Example of a 1024-Entry Receive Connection Table

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

ATM channels from service type VBR, UBR+, GBR, or when Scalable mode is enabled, have an extension to their TCT table entry which is the channel's connection table extension (TCTE) entry. See Section 30.3.2, "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.)

30.3.4.1 Receive Connection Table (RCT)

Figure 30-30 shows the format of an RCT entry.

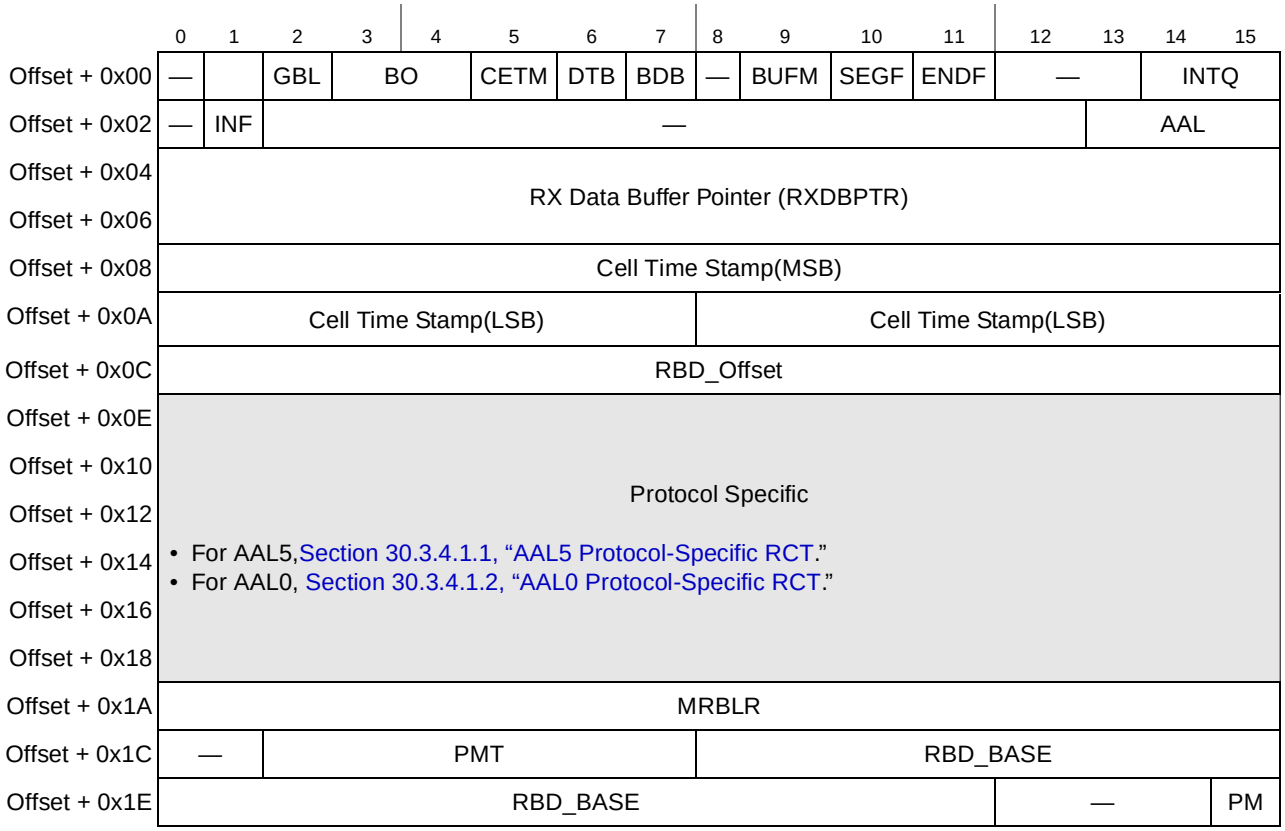


Figure 30-30. Receive Connection Table (RCT) Entry

Table 30-31 describes RCT fields.

Table 30-31. RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0	—	Reserved, should be cleared.
	1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, "Serial DMA Mode Register (SDMR)."
	3–4	BO	Byte ordering—used for data buffers. 00, 01, 11 Reserved 10 Big endian
	5	CETM	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the "Debug Configuration" subsection in your device reference manual.
	6	DTB	Data buffers bus 0 Data buffers reside on the coherent system bus. 1 Data buffers reside on the CE secondary bus.
	7	BDB	BD, interrupt queues, free buffer pool and external SRTS logic bus 0 Reside on the coherent system bus. 1 Reside on the CE secondary bus. Note that when using AAL5 in UDC mode, BDs must be placed on the same bus (RCT[DTB] = RCT[BDB]). This is necessary because in UDC mode the user-defined header, which is part of the cell data, is read using the same bus configuration (byte ordering and bus type) as the payload. Therefore, if data is placed on the CSB bus and the BD on the CE secondary bus, the SDMA accesses the UDC header on the CSB bus with the address of the CE secondary bus.
	8	—	Reserved, should be cleared.
	9	BUFM	Buffer mode. (AAL5 only) See 30.3.9.3/30-106. 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.

Table 30-31. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
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.
	2–12	—	Internal use only. Should be cleared.
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 010 AAL5—ATM adaptation layer 5 protocol All others reserved.
0x04	—	RxDBPTR	Receive data buffer pointer. Holds real address of current position in the Rx buffer.
0x08	0–23	Cell Time Stamp	Used for reassembly time-out. Whenever a cell is received, the QUICC Engine time stamp timer is sampled and written to this field. See Section 19.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)." **Note**: In AAL5 8 LSb bits are always zero meaning the granularity of measurement is every 256 timer ticks.
0x0B	—	—	Reserved, This byte is zero.
0x0C	—	RBD_Offset	RxBD offset from RBD_BASE. Points to the channel's current BD. User-initialized to 0; updated by the QUICC Engine block.
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.

30.3.4.1.1 AAL5 Protocol-Specific RCT

Figure 30-31 shows the AAL5 protocol-specific area of an RCT entry.

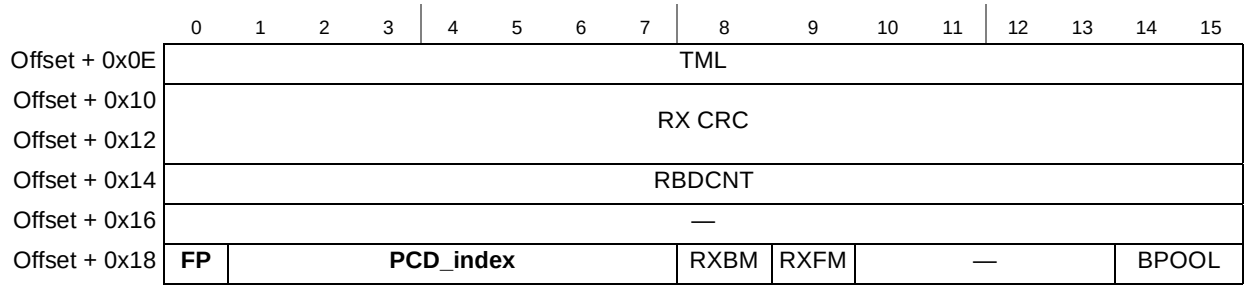


Figure 30-31. AAL5 Protocol-Specific RCT

Table 30-32 describes AAL5 protocol specific RCT fields.

Table 30-32. RCT Settings (AAL5 Protocol-Specific)

Offset	Bits	Name	Description
0x0E	—	TML	Total message length. This field is used by the QUICC Engine block.
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 QUICC Engine block opens a new buffer.
0x16	—	—	QUICC Engine block internal use. Reserved, should be cleared.
0x18	0–8	—	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 30.3.9.2.4/30-104.

30.3.4.1.2 AAL0 Protocol-Specific RCT

Figure 30-32 shows the layout for the AAL0 protocol-specific RCT.

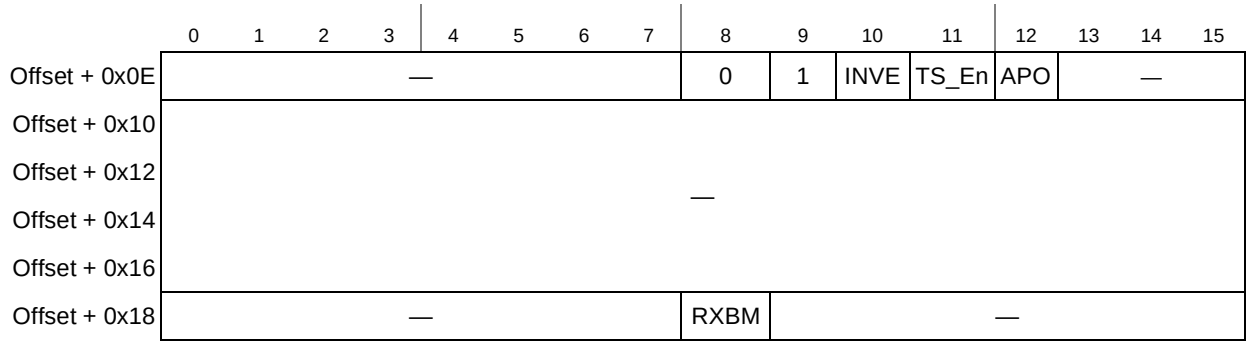


Figure 30-32. AAL0 Protocol-Specific RCT

Table 30-33 describes AAL0 protocol specific RCT fields.

Table 30-33. 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	TS_En	Time stamp Enable. 0 Time stamp is disabled. 1 The QUICC Engine block adds time stamp information to the end of the buffer (just after the end of the cell payload) which is related to this RCT. Useful for OAM support, when it is needed to sample the time of the received OAM cell. It is important that the user will allocate 4 extra bytes to the buffer, in order to use this feature properly.
	12	APO	ATM Payload Only. 0 All the cell including its header and UDC, if applicable, is written to the buffer. 1 Only the cell payload is written to the buffer.
	13–15	—	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.

30.3.4.2 Transmit Connection Table (TCT)

Figure 30-33 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		CETM	DTB	BDB	AVCF	—	ATT		CPUU	VCON	INTQ	
Offset + 0x02	—	INF	—											AAL		
Offset + 0x04 Offset + 0x06	Tx Data Buffer Pointer (TXDBPTR)															
Offset + 0x08	TBDCNT															
Offset + 0x0A	TBD_OFFSET															
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															
Offset + 0x10 Offset + 0x12 Offset + 0x14	Protocol Specific • For AAL5,Section 30.3.4.2.1, “AAL5 Protocol-Specific TCT.” • For AAL0, Section 30.3.4.2.2, “AAL0 Protocol-Specific TCT.”															
Offset + 0x16	APC Linked Channel (APCLC)															
Offset + 0x18 Offset + 0x1A	ATM Cell Header (VPI,VCI,PTI,CLP)															
Offset + 0x1C	—		PMT						TBD_BASE[8–15]							
Offset + 0x1E	TBD_BASE[16–27]												BNM	STPT	IMK	PM

Figure 30-33. Transmit Connection Table (TCT) Entry

Table 30-34 describes general TCT fields.

Table 30-34. TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0	—	Reserved, should be cleared.
	1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
	3–4	BO	Byte ordering. This field is used for data buffers. 00, 01, 11 Reserved 10 Big endian
	5	CETM	QUICC Engine block transaction mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the “Debug Configuration” subsection in your device reference manual.

Table 30-34. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x00	6	DTB	Data buffer bus 0 Resides on the coherent system bus. 1 Resides on the CE secondary bus.
	7	BDB	BD, interrupt queue and external SRTS logic bus 0 Resides on the coherent system bus. 1 Resides on the CE secondary bus. **Note:** When using AAL5 in UDC mode, BDs and data should be placed on the same bus (TCT[DTB]=TCT[BDB]).
	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 QUICC Engine block clears TCT[VCON]. **Note:** When over-subscribing UBR or UBR+ channels, set AVCF so that the QUICC Engine block 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. If Scalable APC is enabled, then the user should allocate an additional 4–6 bytes (depend on the alignment) for the APC scheduling table. See Section 30.3.6.3/30-85, for more information. 11 Guaranteed Bit rate mode (GBR traffic). The host must initialize PCR & PCR fraction fields in this TCT and in the corresponding TCTE. For detailed description see 30.3.4.3/30-78.
	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 (status)	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 QUICC Engine block 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 QUICC Engine block clears VCON.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 30-34. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
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.
	2–12	—	Internal use only. Should be cleared.
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 010 AAL5—ATM adaptation layer 5 protocol 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. For GCRA scheduler see 30.2.12.1/30-21 for details.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. For GCRA scheduler this field is programmed in time units determined by the Time Stamp register. see 30.2.12.1/30-21 for details.
0x10	—	—	Protocol-specific
0x16	—	APCLC	APC mode: APC linked channel. Used by the QUICC Engine block. Initialize to 0 (null pointer). GCRA mode: Must be set to 0xFFFF.
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.

Table 30-34. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
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 [8–27]	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 QUICC Engine block deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. **Note1** 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). **Note2** When working in GBR mode, The sequence of stopping a channel is as follows: First the CBR priority takes the channel from the scheduling table and indicates to the UBR priority level to stop rescheduling the channel on the next appearance of it in the APC. At this point the TCT[VCON] bit is cleared. **Note3** When the channel is a multi-cast channel for example, TCT[MC]=1 there is a special host command for removing a channel from the multicast group. The s/w should not set this bit for stopping such a channel.
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.

30.3.4.2.1 AAL5 Protocol-Specific TCT

Figure 30-34 shows the AAL5 protocol-specific TCT.

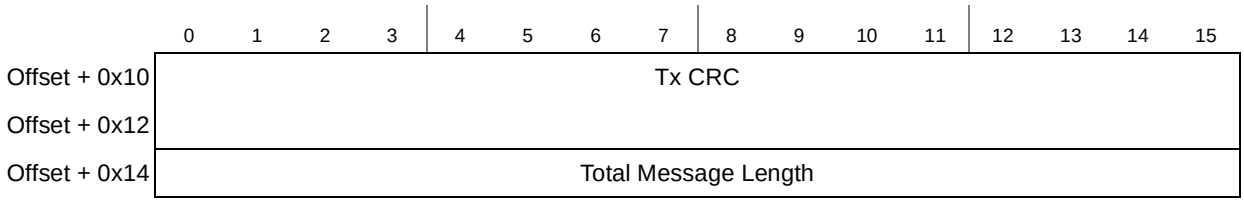


Figure 30-34. AAL5 Protocol-Specific TCT

Table 30-35 describes AAL5 protocol-specific TCT fields.

Table 30-35. AAL5-Specific TCT Field Descriptions

Offset	Name	Description
0x10	Tx CRC	CRC32 temporary result.
0x14	Total Message Length	This field is used by the QUICC Engine block.

30.3.4.2.2 AAL0 Protocol-Specific TCT

Figure 30-35 shows the AAL0 protocol-specific TCT.

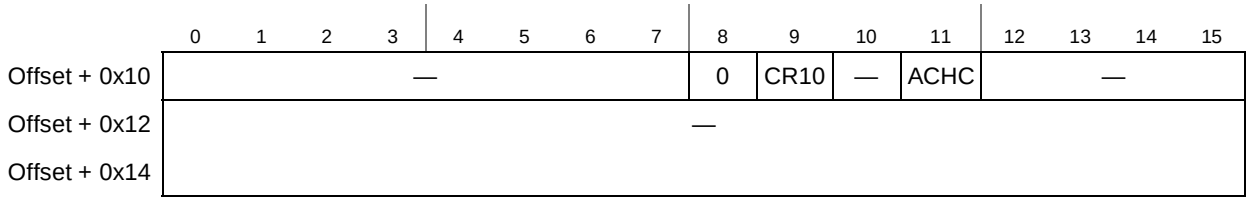


Figure 30-35. AAL0 Protocol-Specific TCT

Table 30-36 describes AAL0 protocol-specific TCT fields.

Table 30-36. 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.

30.3.4.2.3 VBR Protocol-Specific TCTE

Figure 30-36 shows the VBR protocol-specific TCTE.

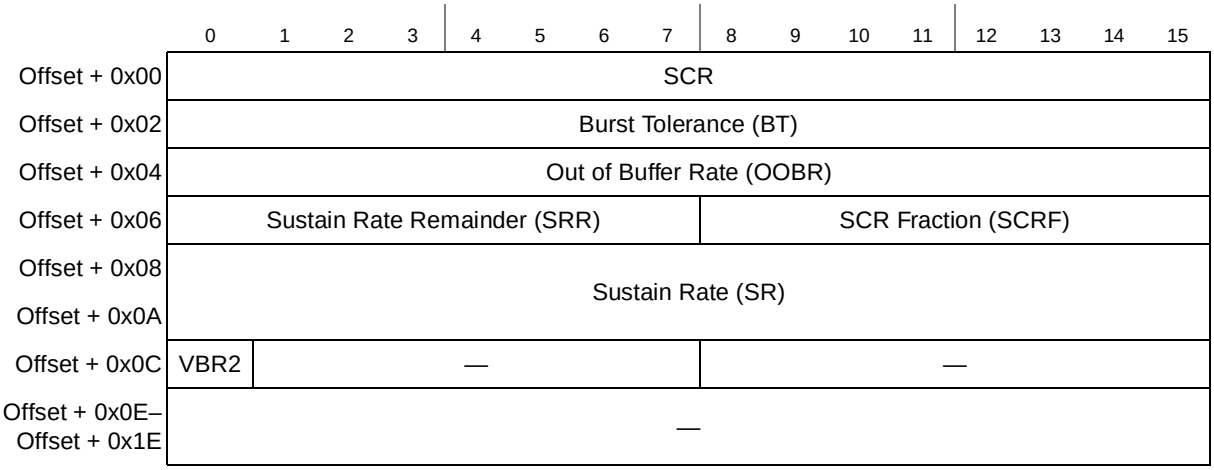


Figure 30-36. Transmit Connection Table Extension (TCTE)—VBR Protocol-Specific

Table 30-37 describes VBR protocol-specific TCTE fields.

Table 30-37. 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). For GCRA scheduler this field is programmed in time units determined by the Time Stamp register. see Section 30.2.12.1, "GCRA Scheduler Rate Programming," for details.
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. For GCRA scheduler this field is programmed in time units determined by the Time Stamp register. see Section 30.2.12.1, "GCRA Scheduler Rate Programming," for details.
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.

Table 30-37. VBR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
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.
	1–7	—	Reserved, should be cleared.
	8–15	—	Reserved, should be cleared.
0x0E–0x1E	—	—	Reserved, should be cleared.

30.3.4.2.4 UBR+ Protocol-Specific TCTE***

Figure 30-37 shows the UBR+ protocol-specific TCTE.

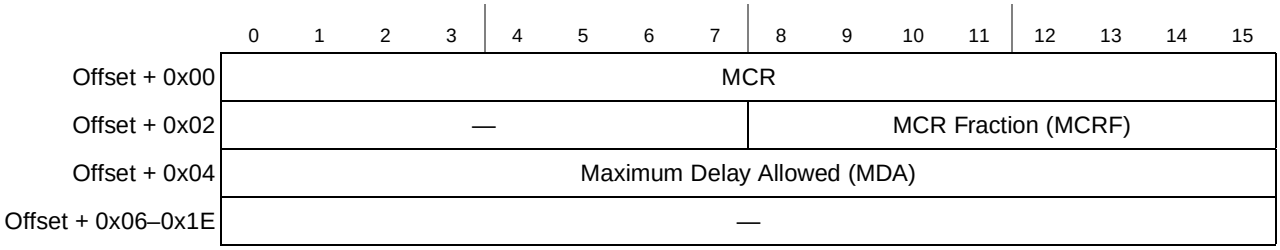


Figure 30-37. UBR+ Protocol-Specific TCTE

Table 30-38 describes UBR+ protocol-specific TCTE fields.

Table 30-38. 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. For GCRA scheduler this field is programmed in time units determined by the Time Stamp register. see Section 30.2.12.1, “GCRA Scheduler Rate Programming,” for details.
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.

30.3.4.2.5 Scalable-APC TCTE

Figure 30-38 shows the Scalable-APC TCTE.

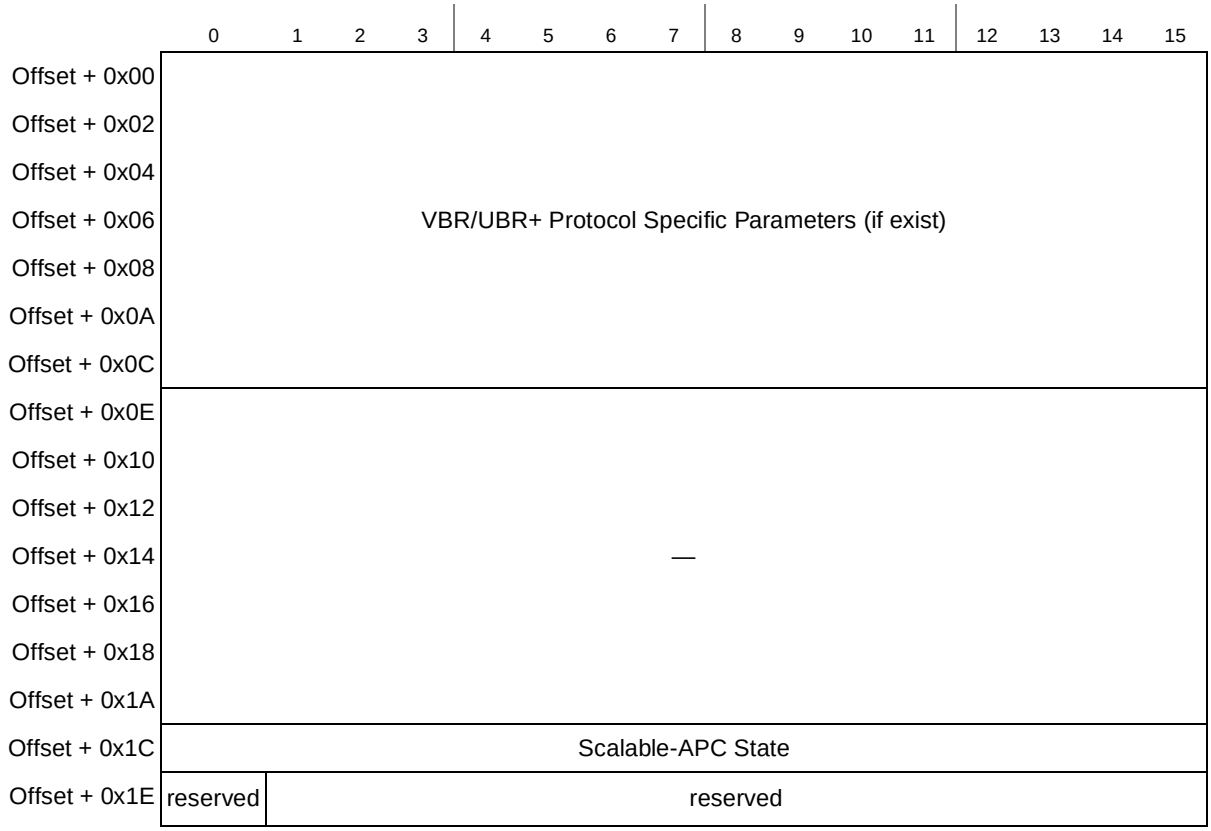


Figure 30-38. Scalable-APC TCTE

Table 30-39 describes Scalable-APC TCTE fields.

Note: This mode does not allow working in GBR- Guaranteed Bit Rate mode described in the Section 30.3.4.3, "GBR Programming Model."

Table 30-39. Scalable-APC TCTE Field Descriptions

Offset	Bits	Name	Description
0x00–0x0c	—	VBR/UBR+ Protocol Specific Parameters	VBR/UBR+ Protocol Specific Parameters (if exist).
0x0E–0x1A	—	—	Reserved, should be cleared.
0x1C–0x1D	—	Scalable-APC State	Reserved, should be cleared. When initiating an STPT command, if the channel is restarted again then these bits should be cleared.
0x1E–0x1F	0	—	reserved.
	1–15		reserved.

30.3.4.3 GBR Programming Model

Figure 30-39 depicts the GBR implementation.

An ATM channel is scheduled for transmission on two priority levels. One has a CBR contract residing on the highest priority level and the other has a UBR contract residing on any other lower priority scheduling table. The host should initialize both TCT & TCTE. The TCT contains the parameters which are common to both, the higher priority-CBR and lower priority-UBR. These parameters include the ATM cell header, BD ring information and internal statuses for the channel- (TBD_BASE, ATM Cell Header....) It also contains the CBR traffic parameters: PCR, Rate remainder, PCR fraction and APC Linked Channel. Its TCT[ATT]=0b11 indicating its a GBR type. On the highest priority there is no need to set the control slot described in Section 30.3.6.3, “APC Scheduling Tables” for fetching the TCTE data structure.

On the UBR priority level the control slot should indicate the need for the TCTE data structure as this priority level is configured based on the parameters residing in this data structure. The TCTE contains the traffic parameters: PCR, PCR fraction and the APC Linked Channel on this priority level. It is described in Figure 30-40.

Working in GCRA scheduler requires programming the PHY parameters table and some predefined rules of how to set the channels in this mode. The TCT and TCTE are programmed according to the GCRA scheduler programming model.

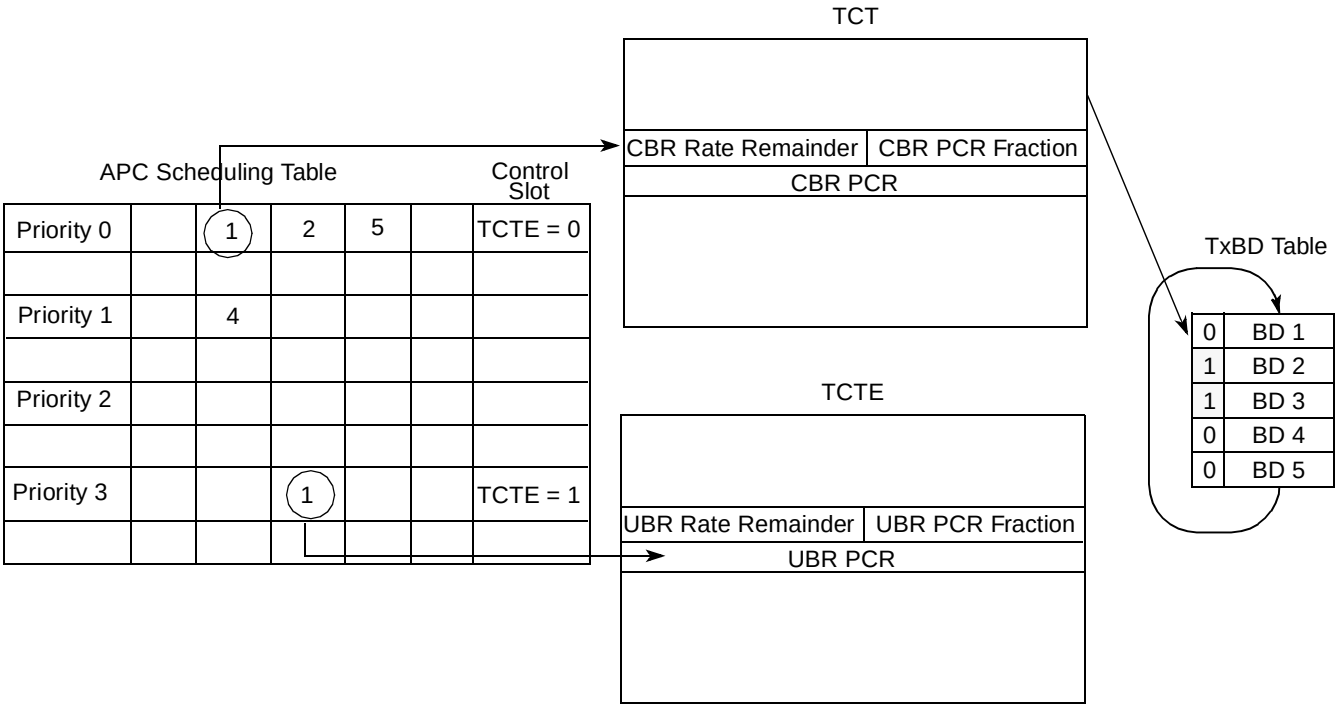


Figure 30-39. GBR Illustration

In termination mode activation of the GBR scheduling is done by two host commands. One for inserting a channel into the higher priority scheduling table and additional command for inserting the same channel into the UBR priority level. See Table 30-65.

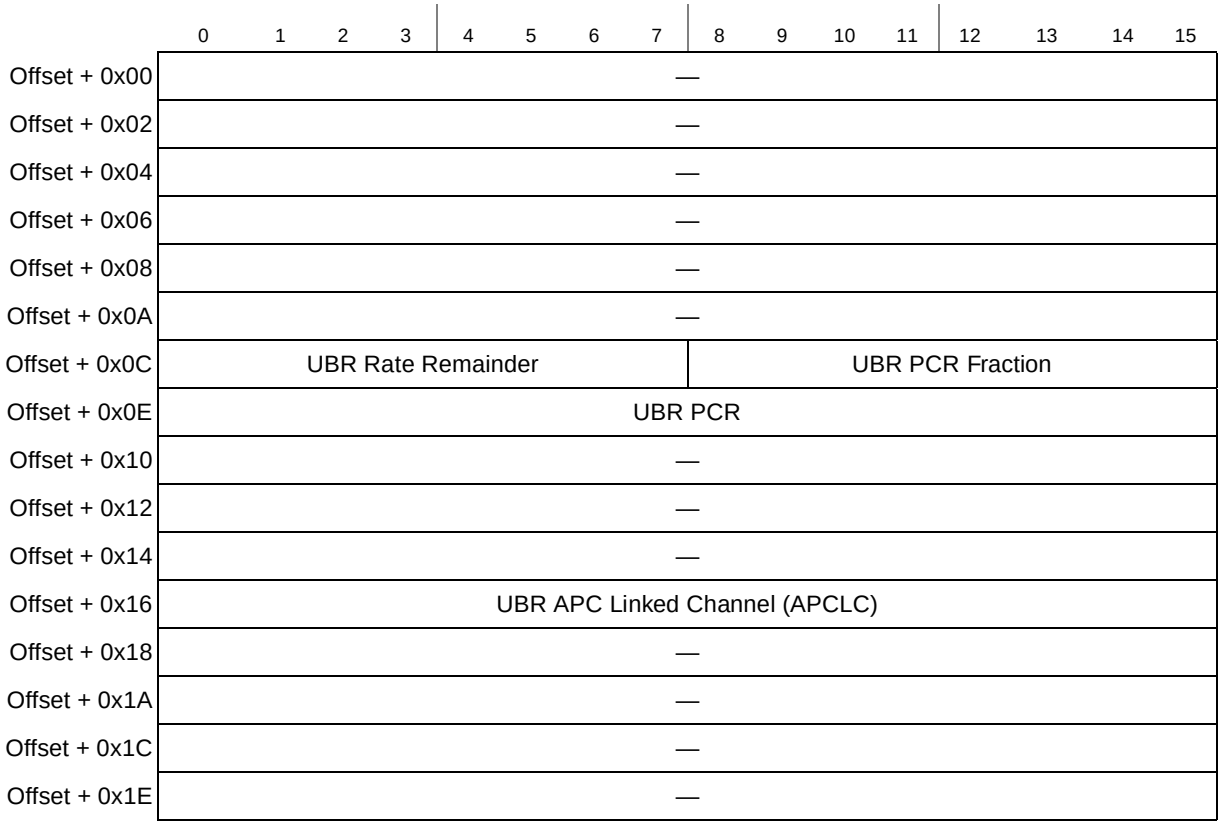


Figure 30-40. GBR Protocol-Specific TCTE

Table 30-40 describes the fields of the GBR protocol-specific TCTE.

Table 30-40. GBR Protocol-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00–0x0A	—	—	Reserved, should be cleared.
0x0C	0–7	**UBR 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	**UBR PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. For GCRA scheduler see Section 30.2.12.1, “GCRA Scheduler Rate Programming” for details.
0x0E	—	**UBR PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. For GCRA scheduler this field is programmed in time units determined by the Time Stamp register. see Section 30.2.12.1, “GCRA Scheduler Rate Programming” for details.
0x10–0x14	—	—	Reserved, should be cleared.
0x16	—	UBR APCLC	APC mode: APC linked channel. Used by the QUICC Engine block. Initialize to 0 (null pointer). GCRA mode: Must be set to 0xFFFF.

Table 30-40. GBR Protocol-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	—	—	Reserved, should be cleared.
0x1A	—	—	Reserved, should be cleared.
0x1C	—	—	Reserved, should be cleared.
0x1E	0–13	—	Reserved, should be cleared.
	14–15	—	Reserved, should be cleared.

The operation of GBR scheduling is as follow:

- Initialize an APC priority table for the highest priority (CBR).
- Initialize an APC priority table for the lower priority (UBR) with bit TCTE in the control slot set.
- Set the TCT[ATT] bits to 0b11.
- Initialize TCT[PCR], TCT[PCR fraction] and TCT[Rate remainder] with CBR related parameters.
- Initialize TCTE[PCR], TCTE[PCR fraction], TCTE[Rate remainder] with UBR related parameters.
- Issue an ATM transmit command related to CBR priority traffic:

  The PRI field in the COMM_INFO should be reset (highest priority) and ACT field should be reset too. This puts the channel into the highest priority on the APC.
- Issue an ATM transmit command for GBR-UBR related to UBR priority traffic. In this command, the channel code is the same as the previous ATM transmit command.

  The PRI field should be different from zero (not highest priority) and ACT field should be 0b10. For GCRA scheduler this should always be the lowest priority level. (For details on GCRA scheduler and GFR mode see Section 30.3.7.1.1, “GBR Operation in GCRA Scheduler Mode.”

30.3.5 OAM Performance Monitoring Tables

The OAM performance monitoring tables include performance monitoring block test parameters, as shown in Figure 30-41. Each block test needs a 32-byte performance monitoring table in the multi-user RAM. In the connection’s RCT and TCT, the user allocates an OAM performance table to a VCC or VPC. See Section 30.2.19, “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.

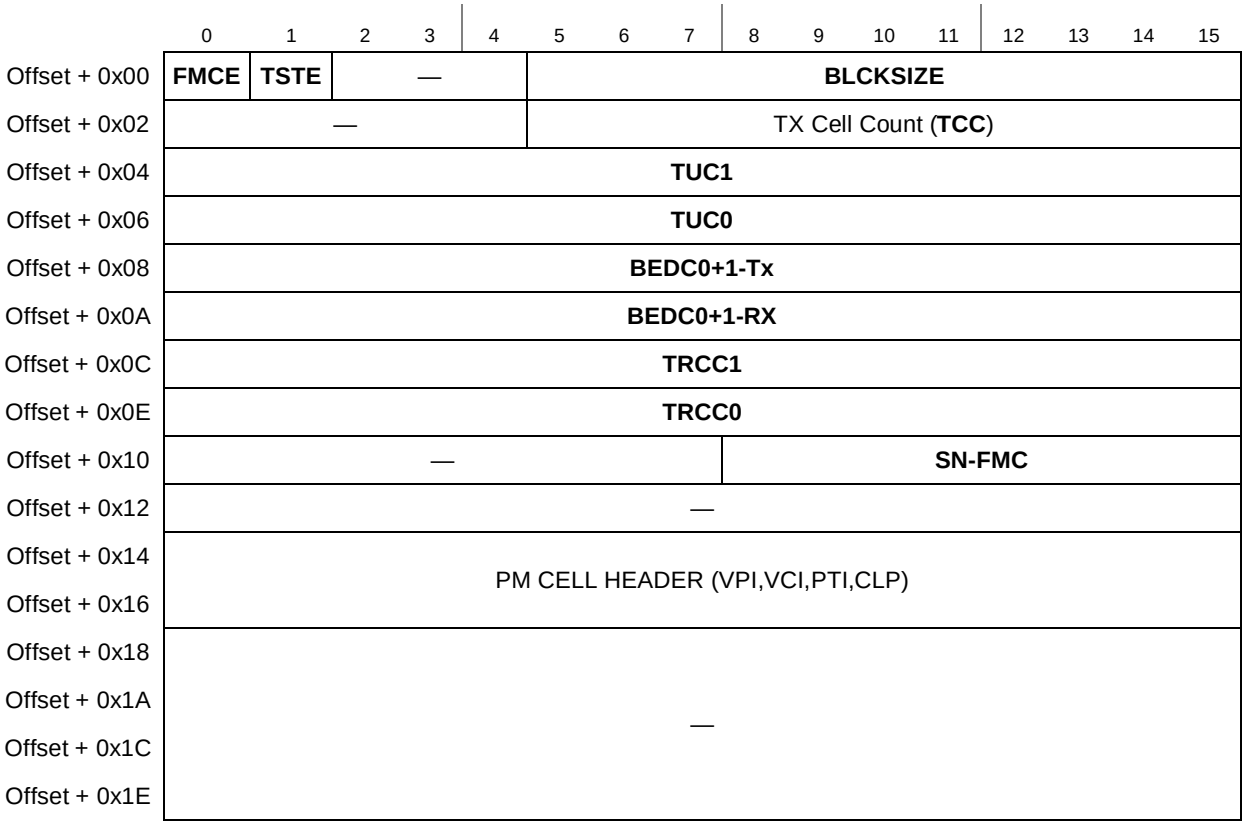


Figure 30-41. OAM Performance Monitoring Table

Table 30-41 describes the fields of the performance monitoring table.

Table 30-41. OAM—Performance Monitoring Table Field Descriptions

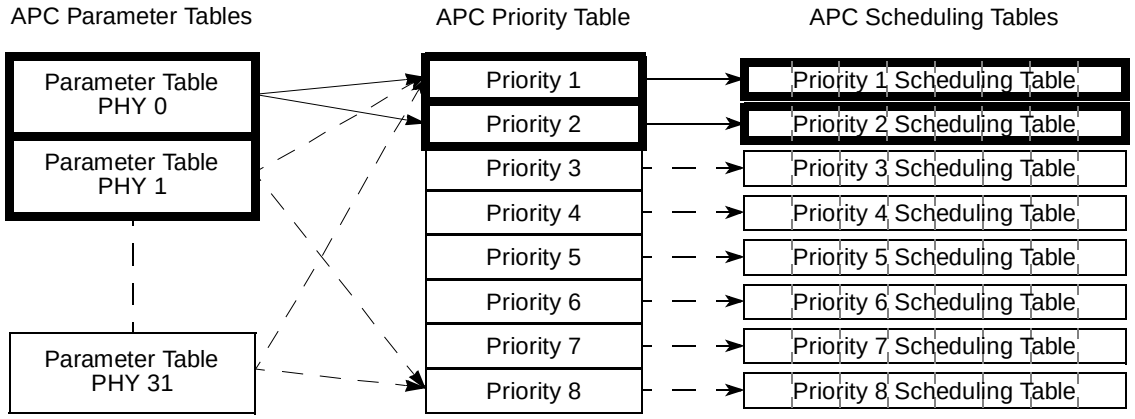
Offset	Bits	Name	Description
0x00	0	**FMCE**	Enables FMC transmission. Initialize to 1.
	1	**TSTE**	FMC time stamp enable 0 The time stamp field of the FMC is coded with all 1's. 1 The value of the time stamp timer is inserted into the time stamp field of the FMC.
	2–4	—	Reserved, should be cleared.
	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 QUICC Engine block 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.

Table 30-41. OAM—Performance Monitoring Table Field Descriptions (continued)

Offset	Bits	Name	Description
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 QUICC Engine block into the Tx cell flow.
0x18–0x1E	—	—	Reserved, should be cleared.

30.3.6 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 30-42.



Note: The highlighted areas represent the active structures for an example implementation of PHY 0 with two priorities. (The unhighlighted areas and dashed arrows represent unused structures.)

Figure 30-42. ATM Pace Control Data Structure

30.3.6.1 APC Parameter Tables

Each PHY's APC parameter table, shown in Table 30-42, holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The table resides in the multi-user RAM. The parameter APCP_BASE, described in Section 30.3.2, "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. For multi device situations the device holds the MSB part of the PHY ID such that device 0 occupies PHY's 0–31, device 1 occupies PHY's 32–63 etc. For ATM MPHY Slave mode, the APC PHY parameter table is located only on the Last PHY location.

Table 30-42. 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 QUICC Engine block. User-initialized with APCL_FIRST.
0x06	CPS	Byte	**Note:** Cells per slot. Determines the number of cells sent per APC slot. See Section 30.2.5.2, "APC Unit Scheduling Mechanism." User-defined. (0x01 = 1 cell; 0xFF = 255 cells.)
0x07	CPS_CNT	Byte	Cells sent per APC slot counter. User-initialized to CPS; used by the QUICC Engine block.
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	—	Byte	Reserved. Should be cleared.
0x0A	FIRST_UBR+ _LEVEL	Hword	This field contains the address of the first UBR+ Priority level. For example, if the first UBR+ priority level is 4 then the address should be calculated as AFP_FIRST + 8 x (UBR+ priority level number- 1). See Section 30.2.10, "Prioritized UBR+ Mode," for more information.
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.
0x14	—	Word	Reserved for QUICC Engine block use. Should be cleared.
0x18	APC_SLOT_DUR _VAL_INT	Word (31 bits)	APC Slot Duration Value Integer. Valid only when Scheduler_MODE[AFC] is set, See Table 30-43. Initialized by the user and it represent the expected time duration of each APC slot. The value programmed here should reflect the maximum rate of the selected PHY. i.e the minimum time required to the APC to send CPS cells. See description in Section 30.2.8, "APC Flux Compensation."
0x1C	APC_SLOT_DUR _VAL_Frac	HWord (16 bits)	APC Slot Duration Value Fraction. Valid only when Scheduler_MODE[AFC] is set, See Table 30-43 Initialized by the user and it represent the fraction of the expected time duration of each APC slot, See description in Section 30.2.8, "APC Flux Compensation."
0x1E	Scheduler_MODE	Hword	Defined in Table 30-43.

[1] Offset values are to APCP_BASE+PHY# × 32. However, in slave mode, the offset is from APCP_BASE regardless of the PHY address.

30.3.6.1.1 Scheduler_MODE

Figure 30-43 shows the layout of the Scheduler_MODE.

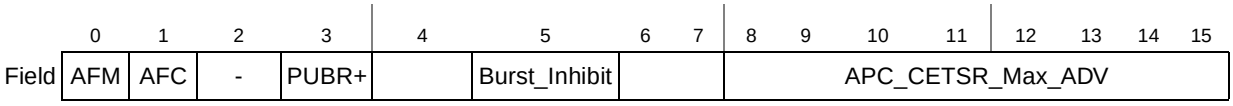


Figure 30-43. Scheduler_MODE

Table 30-43 describes Scheduler_MODE fields.

Table 30-43. Scheduler_MODE Field Descriptions

Bits	Name	Description
0	—	Reserved. Should be cleared.
1	AFC	APC Flux Compensation mode. 0 - No time compensation mode. 1 -APC Time Compensation mode is enabled. See Section 30.2.8, “APC Flux Compensation.”
2	—	Reserved. Should be cleared.
3	PUBR+	Prioritized UBR+ mode. 0 - Prioritized UBR+ mode is disabled. 1 - Prioritized UBR+ mode is enabled. See Section 30.2.10, “Prioritized UBR+ Mode,” for more information.
4	—	Reserved. Should be cleared.
5	Burst_Inhibit	APC Burst Inhibit. Valid for Flux or traditional APC modes. 0 - APC Burst Inhibit is disabled. This means that theoretically there could be a situation in which the gap between the rptr and the sptr is too big, that even after rescheduling the channel will be allocated still before the rptr. This could lead to the case of burst traffic in line rate for this channel, assuming for example that it is the only channel in the APC scheduling table. 1 - APC Burst Inhibit is enabled. This means that burst traffic is eliminated, by avoiding reschedule of a channel before the RPTR. In this case, the channel will be reschedule to the RPTR itself.
6–7	—	Reserved. Should be cleared.
8–15	APC_CETSR_Max_ADV	APC Maximum CETSR Advanced value. Valid Only if AFC=1. User initialized to the maximum value the RPTRs in the APC table could be advanced. In cases where there is a big delay in the PHY this value sets the limitation of how many APC slots the CETSR should be advanced. It prevents the QUICC Engine block from doing extra calculation and assures proper operation of the APC. In any case this number should not exceed the (number of slots in the APC table-1)

30.3.6.2 APC Priority Table

Each PHY’s APC priority table holds pointers to the APC scheduling table of each priority level. It resides in the multi-user RAM. The priority table can hold up to eight priority levels. Table 30-44 shows the structure of a priority table entry.

Table 30-44. 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	

30.3.6.3 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 30-44. Each slot is a half-word entry. Note that the APC scheduling tables should be cleared before the APC unit is enabled.

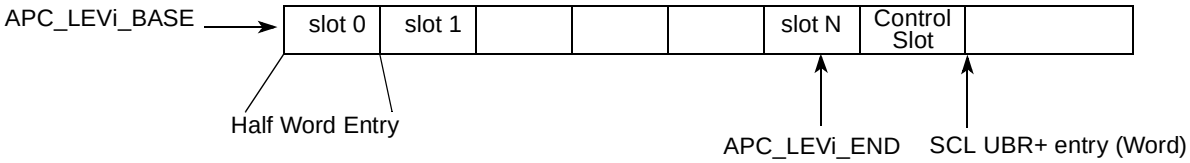


Figure 30-44. The APC Scheduling Table Structure

If working in APC Scalable mode and there are UBR+ channels in the APC table (SCL_UBR bit is set in the control slot see Figure 30-45) the user should allocate an extended control slot which consists of additional 4 or 6 bytes (besides the Control slot) to the APC scheduling table, This is for QUICC Engine block internal use only (should be initialized to 0). The requirement for the last 4 bytes to be word aligned so that the user should allocate additional 4 bytes for this entry if the control slot is at a Non-Word (Word is 4 bytes) aligned address, or allocate an addition of 6 bytes for this entry if the control slot is at a Word aligned address. In these cases the Ext_CS bit should be set see Figure 30-45

Slot N+1 is used as a control slot, as shown in Figure 30-45.

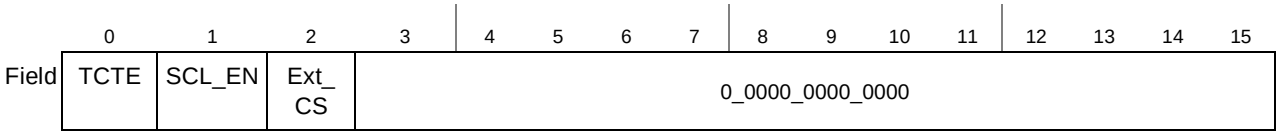


Figure 30-45. Control Slot

Table 30-45 describes control slot fields.

Table 30-45. Control Slot Field Description

Bits	Name	Description
0	TCTE	Used for external channels only. 0 Channels in this scheduling table do not use external TCTE. (None of: external VBR, UBR+, GBRor Scalable-APC channels) 1 Channels in this scheduling table use external TCTE. (External VBR, UBR+, GBRor Scalable-APC channels)
1	SCL_EN	Scalable APC mode. 0 Scalable APC mode is disabled. 1 Scalable APC mode is enabled. This requires clearing offsets 0x1C–0x1F in the TCTE tables of all the channels which resides in this scheduling table. See 30.2.7/30-15.
2	Ext_CS	Extended Control Slot. 0 No need for the extended size control slot. 1 SCL_EN mode is enabled and UBR+ channels for this APC priority table is enabled.
3–15	—	Reserved, should be cleared.

30.3.6.4 UBR+ Priority Decision Table Entry

When the UBR+ priority mode is enabled, the UBR+ priority decision table must be initialized, see Section 30.2.10, “Prioritized UBR+ Mode”. The UBR+ priority decision table is located at address Scheduler_Parameter_Table [AFP_LAST]+8, and each entry contains 4 bytes. Figure 30-46, describes the UBR+ priority decision table entry.

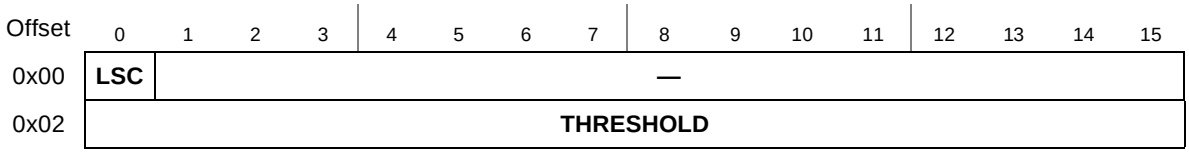


Figure 30-46. UBR+ Priority Decision Table Entry

Table 30-46. UBR+ Priority Decision Table Entry

Offset	Bit	Name	Description
0x00	0	—	Reserved. Should be initialized to zero.
	1–15	Reserved	Reserved. Initialize to zero.
0x02	0–15	THRESHOLD	This field contain the UBR+ MDA value for this priority level (In APC slot units).

30.3.7 GCRA Scheduler Structures

30.3.7.1 GCRA Scheduler PHY Parameter Table

The GCRA scheduler PHY Parameter RAM table occupies 32 bytes, see Figure 30-47. The user can configure each PHY to work in traditional APC or GCRA scheduler mode, by setting GCRA_SCHED bit. For more explanation about the GCRA scheduler mode see Section 30.2.12, “GCRA Scheduler”.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
offset + 0 offset + 2	Time Stamp Shadow Register0															
offset + 4 offset + 6	Time Stamp Shadow Register1															
offset + 8 offset + A	Time Stamp Shadow Register2															
offset + C offset + E	Time Stamp Shadow Register3															
offset + 10	**GCRA_SCHED = 1**	**TCTE**		**Expand**	**GBR_En**		**—**		**Last_priority**				**Size**			
offset + 12	**FIRST_CC**															
offset + 14	**Res**	**Res**		**Res**	**Res**		**—**									
offset + 16	**GCRA_SCHED_Table_ptr**															
offset + 18	**Res**															
offset + 1A	**Res**															
offset + 1C	**Res**															
offset + 1E	**Res**															

Figure 30-47. GCRA Scheduler PHY Parameter Table

Table 30-47. GCRA Scheduler PHY Parameter Table Description

Offset [1]	Bits	Name	Width	Description
0x00	—	Time Stamp Shadow Register0	Word	Reserved.Should be cleared.
0x04	—	Time Stamp Shadow Register1	Word	Reserved.Should be cleared.
0x08	—	Time Stamp Shadow Register2	Word	Reserved.Should be cleared.
0x0c	—	Time Stamp Shadow Register3	Word	Reserved.Should be cleared.

Table 30-47. GCRA Scheduler PHY Parameter Table Description (continued)

Offset [1]	Bits	Name	Width	Description
0x10	0	**GCRASCHED**	—	GCRASCHED- Should be set. 0 - This PHY work using traditional APC. 1 - This PHY work using GCRA scheduler mode.
	1	**TCTE**	—	Used for external channels only. 0 Channels in this scheduling table do not use external TCTE. (No external VBR, UBR+, GBR) 1 Channels in this scheduling table use external TCTE. (External VBR, UBR+, GBR)
	2	**Expand**	—	Expand Mode- For better memory utilization and smaller memory footprint of the APC scheduling table the default maximum number of ATM channels supported for this APC is 32. In cases where the maximum number of channels in the APC is >= 32 the size of this table is increased by 0x10 bytes. See Table 30-48 and Table 30-49 for details 0- Normal mode support for 32 channels. 1- Expanded size model enabled. Supports max of 64 channels per PHY.
	3	**GBR_En**	—	GBR mode is enabled on this GCRA scheduler. See details in Section 30.3.7.1.1, “GBR Operation in GCRA Scheduler Mode”
	4	**—**	—	Reserved. Should be set to 0.
	5	—	—	Reserved. Should be set to 0.
	6–7	**Last_Priority**	—	The highest Priority Number available for this PHY 00 - Priority 0 - the highest priority. 01 - Priority 1. 10 - Priority 2. 11 - Priority 3.
	8–15	**Size**	—	Size of the comparison table in bytes-1. User initialized to the number of ATM channels which are active in the GCRA scheduling table *2 -1. **Note:** Each GBR channel is considered as two channels for this count.
0x12	—	**FIRST_CC**	Hword	First CC value. User initialized to the first CC that fit to index 0 in the priority x MASK.
0x14	0	**—**	—	Reserved. Should be set to 0.
	1	**—**	—	Reserved. Should be set to 0.
	2	**—**	—	Reserved. Should be set to 0.
	3	**—**	—	Reserved. Should be set to 0.
	4-7	**—**	—	Reserved. Should be set to 0.
	8-31	**GCRA_SCHEDTable_ptr_**	Word	GCRA Scheduler Table pointer. Initialized to the base address of the GCRA scheduler table. Should be aligned to 8 bytes in case of expanded mode or 4 bytes in case of non-expanded mode. The maximum memory allocation for 64 ATM channels in expanded mode is: 32+64*2 = 160 bytes. For a table which has less than 32 channels the size is 16 + Num of channels*2.
0x18	—	—	4*Hword	Reserved. Should be cleared.

[1] Offset from PHY# * 0x20.

Figure 30-48 shows the GCRA scheduling table structure. The user can initialize maximum of 64 channels per PHY, plus extra 32 header bytes for the Pri_MASK entries. Therefore the maximum size of this structure is (32+64*2=) 160 bytes.

This table consists of the theoretical Finish Time (FT) of each of the potential 64 ATM channels codes residing under this PHY. The selection algorithm within this scheduling table is finding the entry which has the minimum Finish time in each priority, starting from the highest priority, and checking if this minimum FT is smaller than the current system time (CETSR). If this is the case then the Channel Code is selected for transmission, and in the rescheduling process the FT will be updated according to the PCR/PCR Fraction values (for example, for CBR channel). On the other hand, if there is no channel for transmission from the highest priority, the QUICC Engine block continues to search a channel from lower priorities, and so on.

30.3.7.1.1 GBR Operation in GCRA Scheduler Mode

If GBR bit is set enabling Guaranteed Bit Rate scheduling, the QUICC Engine block requires a predefined channel code assignment for those GBR channels. The CBR part of this traffic is located in priority 0-identical to the standard APC. The UBR part of this traffic resides in the lowest priority ONLY. In this case the lowest priority is designated for the UBR part of the GBR traffic and should not have any other types of connections assigned to it.The host has to issue two host commands for inserting the same channel into two priority levels. Even though it is a single channel it consumes the space of two channels since each of the traffic priorities requires a Finish Time of its own. Upon issuing the host command for the CBR traffic the channel is inserted and the priority mask is updated according to the channel number. For the UBR priority the QUICC Engine block will automatically assign the next channel number and will set the priority mask accordingly even though there is no TCT/TCTE initialized for this channel number. The UBR priority must reside on the lowest priority level. The channel numbers of the CBR traffic must have even numbers and the UBR channels will have the following number. Once a UBR channel is selected for transmission the QUICC Engine block will fetch the TCT and TCTE of the CBR priority level.

For example if channel #368 is assigned to the CBR traffic, channel #369 should not be used in any other priority levels and is designated for the UBR priority level. TCT and TCTE for channel #369 should not ne initialized.

Figure 30-48. GCRA Scheduling Table Structure - Expand =1

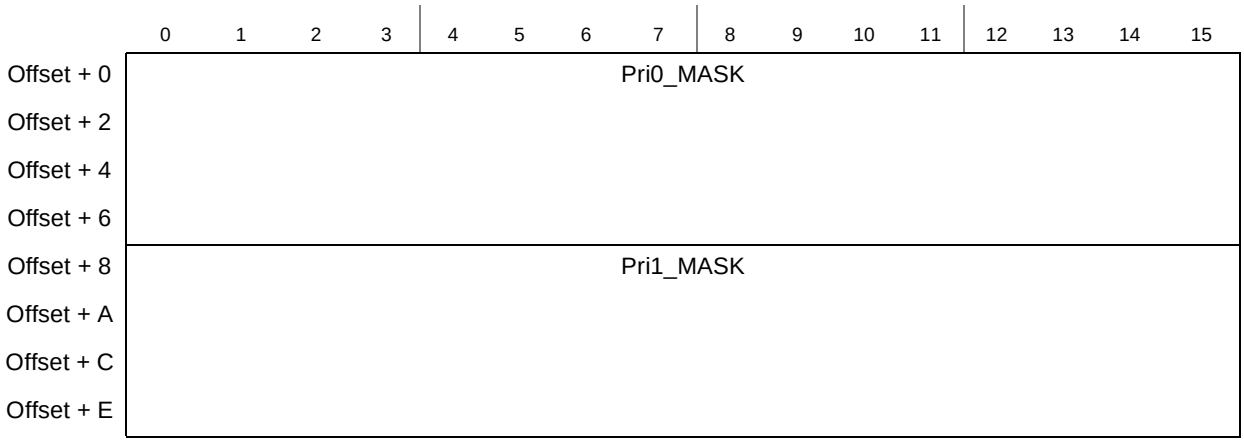


Figure 30-48. GCRA Scheduling Table Structure - Expand =1 (continued)

Offset + 10 Offset + 12 Offset + 14 Offset + 16	Pri2_MASK
Offset + 18 Offset + 1A Offset + 1C Offset + 1E	Pri3_MASK
Offset + 20	CH 0 FT
Offset + 22	CH 1 FT
Offset + 24	CH 2 FT
Offset + 26	CH 3 FT
Offset + 28	CH 4 FT
Offset + 2A	CH 5 FT
Offset + 2C	CH 6 FT
Offset + 2E	CH 7 FT
Offset + 30 - offset + 96	: : :
Offset + 98	CH 60FT
Offset + 9A	CH 61FT
Offset + 9C	CH 62FT
Offset + 9E	CH 63FT

Table 30-48. GCRA Scheduler PHY Scheduling Table Description - Expand=1

Offset	Bits	Name	Width	Description
0x00	—	**Pri0_MASK**	DWord	Priority 0 MASK. The relevant bits that corresponds to the active channels in priority 0 (up to 64 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, “Dynamic Adding and Removing Channels.” If GBR scheduling mode is enabled it implies that two consecutive channels are running, one having an even number- N, on this priority level and the other on the lowest priority level having an odd number N+1.
0x08	—	**Pri1_MASK**	DWord	Priority 1 MASK. The relevant bits that corresponds to the active channels in priority 1 (up to 64 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, “Dynamic Adding and Removing Channels.”

Table 30-48. GCRA Scheduler PHY Scheduling Table Description - Expand=1 (continued)

Offset	Bits	Name	Width	Description
0x10	—	**Pri2_MASK**	DWord	Priority 2 MASK. The relevant bits that corresponds to the active channels in priority 2 (up to 64 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, “Dynamic Adding and Removing Channels”.
0x18	—	**Pri3_MASK**	DWord	Priority 3 MASK. The relevant bits that corresponds to the active channels in priority 3 (up to 64 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, “Dynamic Adding and Removing Channels.”
0x20–0x9F	—	**Ch x FT (x=0..63)**	Hword	Channel Finish Time. Initialized with 0. The QUICC Engine block updates the channel theoretical time for the next transmission. Number of entries to be allocated in the table is determined by the number of active channels in the GCRA.

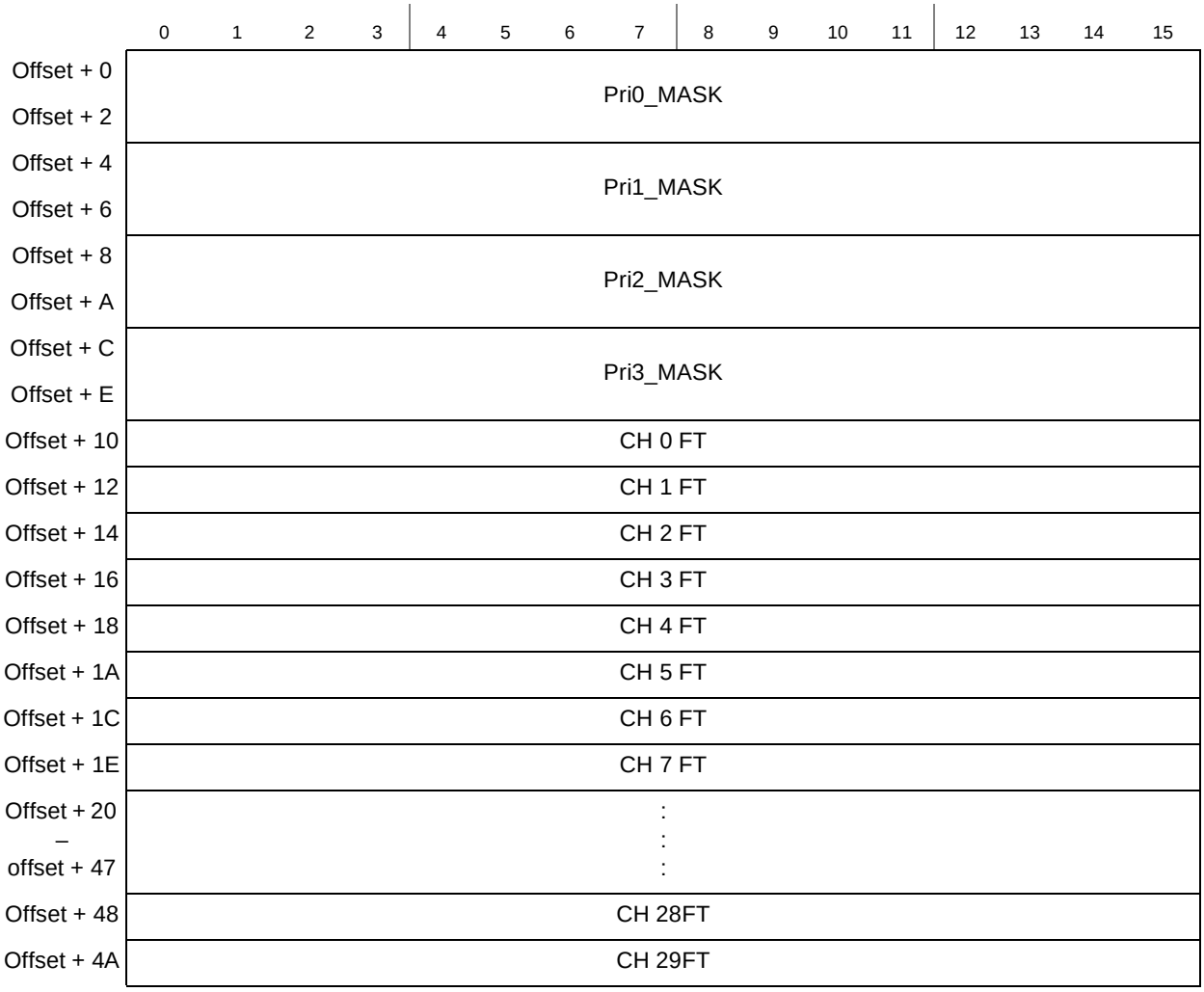


Figure 30-49. GCRA Scheduling Table structure - Expand=0

Offset + 4C	CH 30FT
Offset + 4E	CH 31FT

Figure 30-49. GCRA Scheduling Table structure - Expand=0

Table 30-49. GCRA Scheduler PHY Scheduling Table description- Expand=0

Offset	Bits	Name	Width	Description
0x00	—	**Pri0_MASK**	Word	Priority 0 MASK. The relevant bits that corresponds to the active channels in priority 0 (up to 32 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, "Dynamic Adding and Removing Channels." If GBR scheduling mode is enabled it implies that two consecutive channels are running, one having an even number- N, on this priority level and the other on the lowest priority level having an odd number N+1.
0x04	—	**Pri1_MASK**	Word	Priority 1 MASK. The relevant bits that corresponds to the active channels in priority 1 (up to 32 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, "Dynamic Adding and Removing Channels".
0x8	—	**Pri2_MASK**	Word	Priority 2 MASK. The relevant bits that corresponds to the active channels in priority 2 (up to 32 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, "Dynamic Adding and Removing Channels".
0x0C	—	**Pri3_MASK**	Word	Priority 3 MASK. The relevant bits that corresponds to the active channels in priority 3 (up to 32 channels in all the priorities), are set by the QUICC Engine block. This parameter can be dynamically changed, if the channel is removed or being add to the GCRA. This can be done only by issuing host command. See Section 30.2.12.2, "Dynamic Adding and Removing Channels".
0x20–0x4F	—	**Ch x FT (x=0..31)**	Hword	Channel Finish Time. Initialized with 0. The QUICC Engine block updates the channel theoretical time for the next transmission. Number of entries to be allocated in the table is determined by the number of active channels in the GCRA.

30.3.8 Hierarchical Scheduling

In application where a single ATM VC has to carry a few traffic services with different QOS hierarchical scheduling is available. Selection process is performed per Frame or per cell of up to eight queues per single ATM channel using WFQ algorithm for transmission of AAL5 frames. This mode is depicted in Figure 30-50 and it requires data structures as described in the following paragraphs.

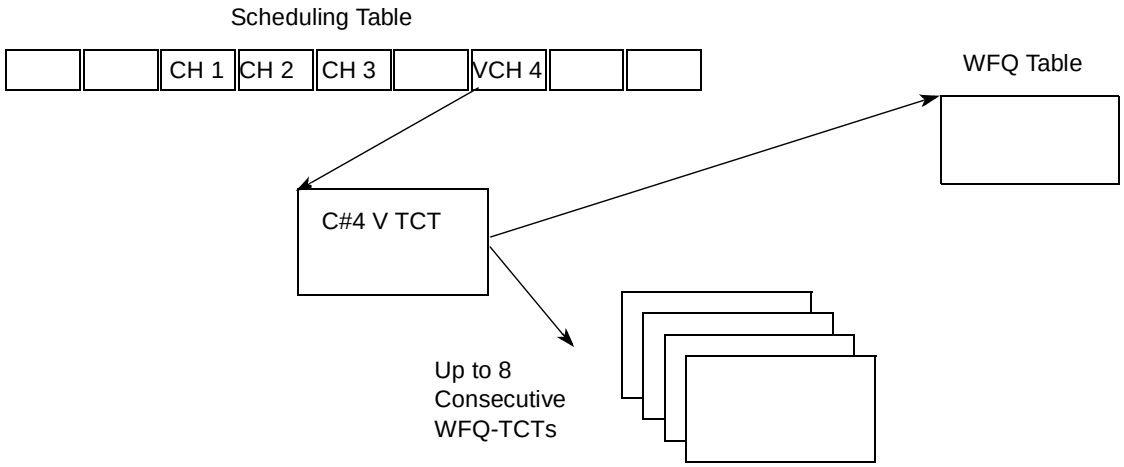


Figure 30-50. Hierarchical Scheduling

In hierarchical scheduling the scheduler schedules a virtual channel with a TCT (V-TCT) which is scheduled in the APC or the GCRA scheduler. This channel's TCT contains management parameters for a set of up to eight queues, each represented by a WFQ -TCT. These queues are represented as ATM channels and are not scheduled in the APC. Their channel numbers should be consecutive and always be assigned such that the first channel code of the eight is divisible by eight. Figure 30-51 describes programming model of the V-TCT Virtual channel TCT which is the root of the hierarchy. In case the channel has a VBR, GBR or UBR+ traffic contract a TCTE should be initialized as well.

The Hierarchical scheduling can operate in two modes of scheduling scheme: HFB- Hierarchical Frame Based and HCB- Hierarchical Cell based scheduling. The mode is determined by each of the WFQ-TCTs which are active under the root V-TCT. In the HFB mode the WFQ selection is performed once every AAL5 frame. Once a frame is transmitted the WFQ algorithm is triggered, the previously selected queue is penalized with the cost function specified in the WFQ data structure multiplied by the frame length and the next queue (WFQ-TCT) is selected for transmission of a complete frame. In HCB mode the WFQ selection is performed after completing transmission of a single ATM cell. In this case the penalty function takes the length value as 48 bytes which is the length of an ATM cell.

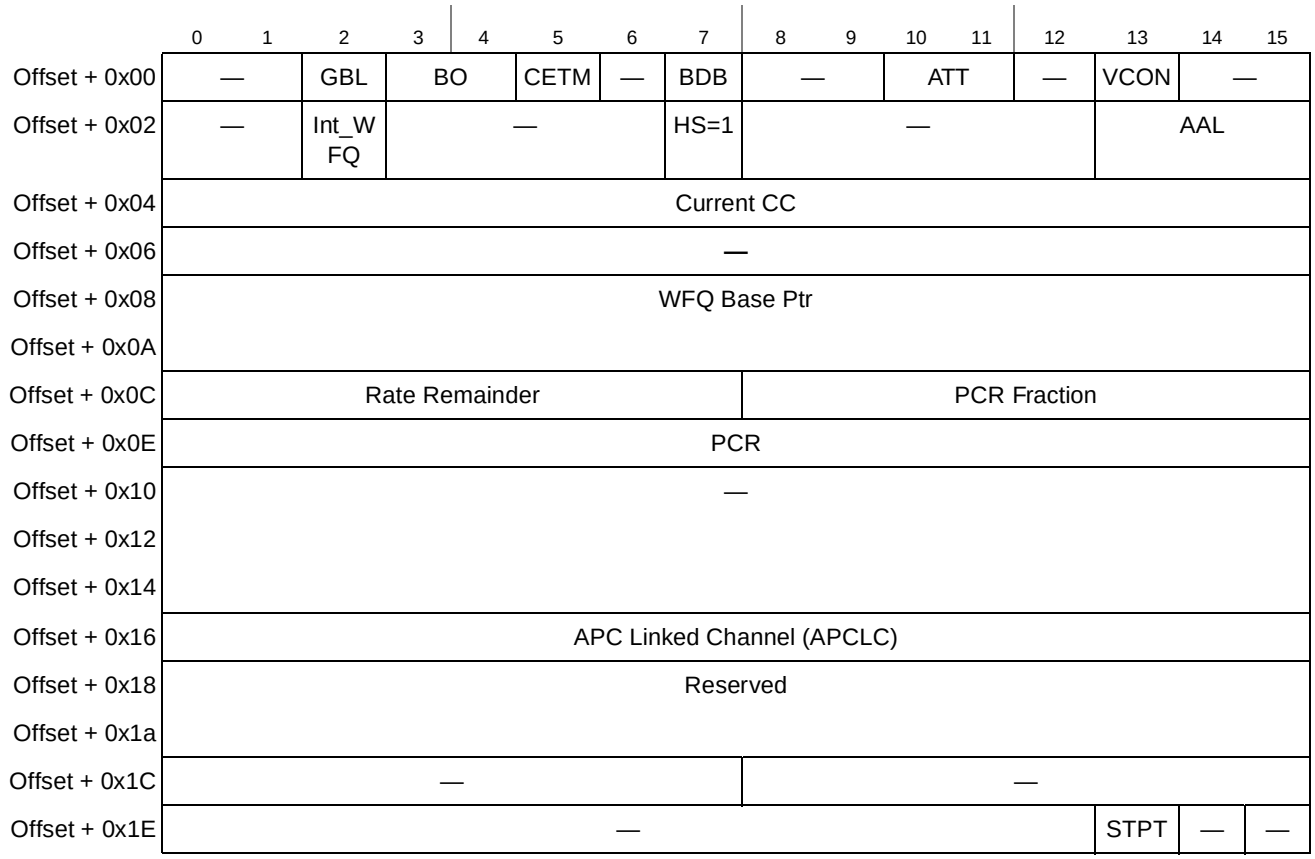


Figure 30-51. Virtual Transmit Connection Table (V-TCT) Entry

Table 30-50 describes the data structure fields.

Table 30-50. V-TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0	—	Reserved, should be cleared.
	1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
	3–4	BO	Byte ordering. This field is used for data buffers. 00, 01, 11 Reserved 10 Big endian

Table 30-50. V-TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x00	5	CETM	QUICC Engine block transaction mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the “Debug Configuration” subsection in your device reference manual.
	6	—	Reserved. Should be cleared
	7	BDB	BDB Should have the identical value to ALL of the WFQ TCTs, as stated in the description of WFQ_Base_Ptr.
	8	—	Reserved. Should be cleared
	9	—	Reserved.
	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. If Scalable APC is enabled, then the user should allocate an additional 4–6 bytes (depend on the alignment) for the APC scheduling table. See Section 30.3.6.3, “APC Scheduling Tables,” for more information. 11 Guaranteed Bit rate mode (GBR traffic). The host must initialize PCR & PCR fraction fields in this TCT and in the corresponding TCTE. For detailed description see Section 30.3.4.3, “GBR Programming Model.”
	12	—	Reserved. Should be cleared
	13	VCON (status)	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 QUICC Engine block 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 QUICC Engine block clears VCON.
	14–15	—	Reserved. Should be cleared
0x02	0–1	—	Reserved. Should be cleared
	2	Int_WFQ	Internal WFQ table. When working in Hierarchical cell based mode it is recommended to locate the WFQ data structure in the MURAM. This will remove accessing external memory on a per cell basis. 0- External WFQ tables. In this case the size of the table is fixed to 32bytes (8 entries) 1- Internal WFQ tables. The WFQ tables reside in the MURAM and the size is determined in V-TCT[WFQ Size]
	3–6		Reserved. Should be cleared
	7	**HS**	Hierarchical scheduling enabled. 0- Not enabled 1- This channel is working in hierarchical frame based mode.
	8–12	—	Reserved. Should be cleared
	13–15	AAL	AAL type. For Frame based mode of operation only AAL5 is operational 010 AAL5—ATM adaptation layer 5 protocol

Table 30-50. V-TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x04	—	**Current CC**	This is the channel code that is currently selected by the WFQ algorithm for transmission. The user initialize this entry to **zero** when activating this channel into scheduling. Once any of the queues in the hierarchy is activated the QUICC Engine block will automatically update this field.
0x06			
0x08	—	**WFQ Base Ptr**	This is a pointer to 32 bytes table in external memory. The bus selection of this data structure is determined by the TCTs of the channels which are selected by this virtual channel. All these channels have their TCT[BDB] identical. Description of the WFQ table is in Figure 30-55.
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 for GCRA scheduler see Section 30.2.12.1, “GCRA Scheduler Rate Programming” for details.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. For GCRA scheduler this field is programmed in time units determined by the Time Stamp register. see Section 30.2.12.1, “GCRA Scheduler Rate Programming” for details.
0x10	—	—	Reserved. Should be cleared.
0x16	—	APCLC	APC mode: APC linked channel. Used by the QUICC Engine block. Initialize to 0 (null pointer). GCRA mode: Must be set to 0xFFFF.
0x18	—	—	Reserved. Should be cleared
0x1C	0–7	—	Reserved. Should be cleared
	8–15	WFQ Size	The size of the WFQ table in bytes minus 1. For example: for 5 WFQ TCTs the size of the WFQ table must be set to 5*4 - 1 = 19 If the table is located in external memory V-TCT[Int_WFQ]=0b0 the value of the WFQ Size should be programmed to 8*4 - 1 = 31
0x1E	0–12	—	Reserved. Should be cleared
	13	STPT	Stop transmit. Should be cleared initially. When the host sets this bit, the QUICC Engine block deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. **Note** The user should set this bit after all the WFQ channels have stopped transmission.
	14–15	—	Reserved. Should be cleared

Up to eight AAL5 TCTs are selected by the V-TCT. These TCT are called WFQ-TCTs and they contain all the channel parameters. The CC numbers of the channels participating in the WFQ selection should be assigned in a such a way that the first is divisible by eight. The data structure is described in Figure 30-52.

As stated, the selection process of the transmit queue could be performed per frame or per cell and it depends on the configuration of each of the WFQ-TCTs. The WFQ TCTs contain the ATM cell header information and so if Cell based scheduling is enabled the WFQ TCTs should have different values in the ATMCH field in order to prevent frame interleaving. If the scheduling is frame based the WFQ TCT could share a common VPI VCI and interleaving is prevented by the scheduler.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—	—	GBL	BO		CETM	DTB	BDB	AVCF		ATT		CPUU	VCON	INTQ	
Offset + 0x02	—	INF	—	—	—	—	—	HS=1	—	—	—	—	HCB	AAL		
Offset + 0x04	Tx Data Buffer Pointer (TXDBPTR)															
Offset + 0x06																
Offset + 0x08	TBDCNT															
Offset + 0x0A	TBD_OFFSET															
Offset + 0x0C	**WFQ INC**															
Offset + 0x0E	**Virtual CC**															
Offset + 0x10	Protocol Specific • For AAL5,see Section 30.3.4.2.1, “AAL5 Protocol-Specific TCT.”															
Offset + 0x12																
Offset + 0x14																
Offset + 0x16	**WFQ_Default_Length**															
Offset + 0x18	ATMCH- ATM Cell Header (VPI,VCI,PTI,CLP)															
Offset + 0x1a																
Offset + 0x1C			PMT						TBD_BASE[8–15]							
Offset + 0x1E	TBD_BASE[16–27]												BNM	STPT	IMK	PM

Figure 30-52. WFQ Transmit Connection Table (TCT) Entry

This WFQ- TCT is very similar to a standard AAL5 TCT. It is marked as HS- Hierarchical scheduling and differ from the standard TCT in the following entries:

Offset 0x02: Bit **HCB**: Hierarchical **Cell** Based schedule enabled.

When cleared, this WFQ TCT will be operating in Hierarchical Fame Based scheduling- HFB. The queue selection using the WFQ algorithm is triggered after **each frame** transmission.

When set, this WFQ TCT will be operating in Hierarchical Cell Based scheduling. The queue selection using the WFQ algorithm is triggered after **each cell** transmission. (The HS bit should be set as well for this mode). The assumption in this mode is that each of the WFQ TCT has a different VPI VCI associated with it and therefore there is no problem with interleaving of AAL5 frames when channels are transmitted in a mixed manner

Offset 0x0C: **WFQ INC**- This entry is the weight of this queue in the weighted Fair queue algorithm. The bandwidth allocated to the channel is inverse proportional to this number and the frame length transmitted from this queue. For every frame which is transmitted for a selected WFQ TCT the length of the frame is multiplied by this value and the result is accumulated in the WFQ Table described in Figure 30-55 in the entry which associated with the queue. Upon completion of transmitting a frame, the WFQ process selects the next queue for transmission. Selection is based on the queue which has the minimum value stored at the table and which is not empty at the time of the selection.

offset 0x0E:**Virtual CC**- This entry contains the channel number of the V-TCT which is the root of the hierarchy.

offset 0x16: **WFQ Default Length**- Default frame length size, which is used by the QUICC Engine block for the WFQ algorithm. This value is used in case when a BD is not ready for transmission in a middle of a frame and an Abort sequence is triggered. It is multiplied by the **WFQ INC** value which is programmed on each WFQ TCT and it acts as a penalty function. The value is used as a method for defining poling time on which this queue will be revisited on the next time. It is useful for systems without the AVCF (Automatic VC Off) mechanism enabled, since this requires reactivation of the channel by the host.

By setting different values in the WFQ INC field of each of the WFQ TCTs assigned under a V-TCT the application can achieve different scheduling schemes between the eight queues which are assigned to the V-TCT.

For example:

Simple Weighted Fair Queue Scheduling

	WFQ TCT1	WFQ TCT1	WFQ TCT2	WFQ TCT3	WFQ TCT4	WFQ TCT5	WFQ TCT6	WFQ TCT7
WFQ INC	3	5	2	30	2	5	30	3

Figure 30-53. Example of Simple WFQ Among all FIFOs

For simplification of the calculation we assume frame size on each of the queues is identical and therefore has no affect on the result. The formula for getting the bandwidth for each FIFO, assuming FIFO #i has WFQ INC Ki (i=0-7)

FIFO #i BW = line_rate × (1/Ki)/[1/K1+1/K2+..+1/Kn]

For the above example FIFO0 receives 0.16 of the line rate, FIFO1 receives 0.09 and etc. The algorithm implement true WFQ, in the sense that at any given time all FIFOs are examined, and the one with the smallest finish time is chosen.

Strict Priority

Strict priority is a mode in which as long a high priority FIFO has data available for transmission this queue is chosen. Implementation of strict priority among all FIFOs (queue0 is highest, and queue7 is lowest priority) depends on the mode of operation. For systems with no Auto VC Off/ON enabled the application should program all WFQ INC values to a small value, for example 0x01. The value of the **WFQ Default Length** should be set in a way that will not prevent other queues from transmission when there is no data available for higher priority queues. (The value will be big enough to enable other queues to be polled)

Mixed Mode

To implement a mixed mode where one queue has high priority and others are scheduled with WFQs, the strict priority queues should be assigned with the lowest WFQ INC and the other queues are assigned with the relative WFQ INC according to their relative weight. The cost function should be such that the high priority queue will not block the rest of the queues.

	WFQ TCT1	WFQ TCT1	WFQ TCT2	WFQ TCT3	WFQ TCT4	WFQ TCT5	WFQ TCT6	WFQ TCT7
WFQ INC	1	2	3	30	2	5	30	3

Figure 30-54. Example of Mixed Mode WFQ

In this example queue0 will get most of the bandwidth as long as there is data available. Once data is not available the cost function will open a window for other queues for transmission but still pole the highest priority queue. Of the remaining bandwidth the remaining queues are allocated the bandwidth according to their relative weight.

Figure 30-52 depicts the WFQ table associated with each V-TCT. It is pointed from the V-TCT[WFQ Base Ptr]. Each table consists of up to eight entries, each is four bytes. Each entry should be initialized by the host with the value of 0x8000_0000. The MSB of each entry indicates that the queue is Empty. When issuing a host command for a specific queue, the QUICC Engine block will clear the relevant bit for the specific queue and it will take part in the WFQ selection.

Memory allocation for the table varies if it resides in external memory or internal MURAM. See description in V-TCT[WFQ Size]. If external memory is used the data structure is always 32 bytes and should be aligned for this size. In MURAM it should be 4 bytes aligned.

Offset	each entry should be initialized with the value of: 0x80000000
0x00	
0x04	
0x08	
0x0C	
0x10	
0x14	
0x18	
0x1C	

Figure 30-55. WFQ Structure

30.3.8.1 Initialization of Hierarchical Channels

The host should follow this sequence in order to activate hierarchical scheduling.

- Initialize a V-TCT with the desired traffic parameters.
- Initialize the WFQ structure as described in Figure 30-55
- Initialize the WFQ-TCTs taking part in the WFQ selection. according to the numbers of channels
- For WFQ channels which **are not** members in a multicast group perform the following steps:

— Issue a host command for inserting the WFQ channel into the WFQ structure. Repeat this command for each channel participating in the WFQ selection. This is an ATM transmit command with special parameters described in Section 30.4.1, "ATM Commands." The ACT field is set to 0b11 and the Virtual CC is given in the BT field in the command info area.

— Issue an host command for scheduling the V-TCT. This is an ATM transmit command with parameters for inserting the CC into the scheduling table (APC or GCRA scheduler). If the V-TCT is scheduled in GBR mode another host command should be issued for inserting the channel to the UBR level and the Command info[ACT]=0b10.

30.3.9 ATM Controller Buffer Descriptors (BDs)

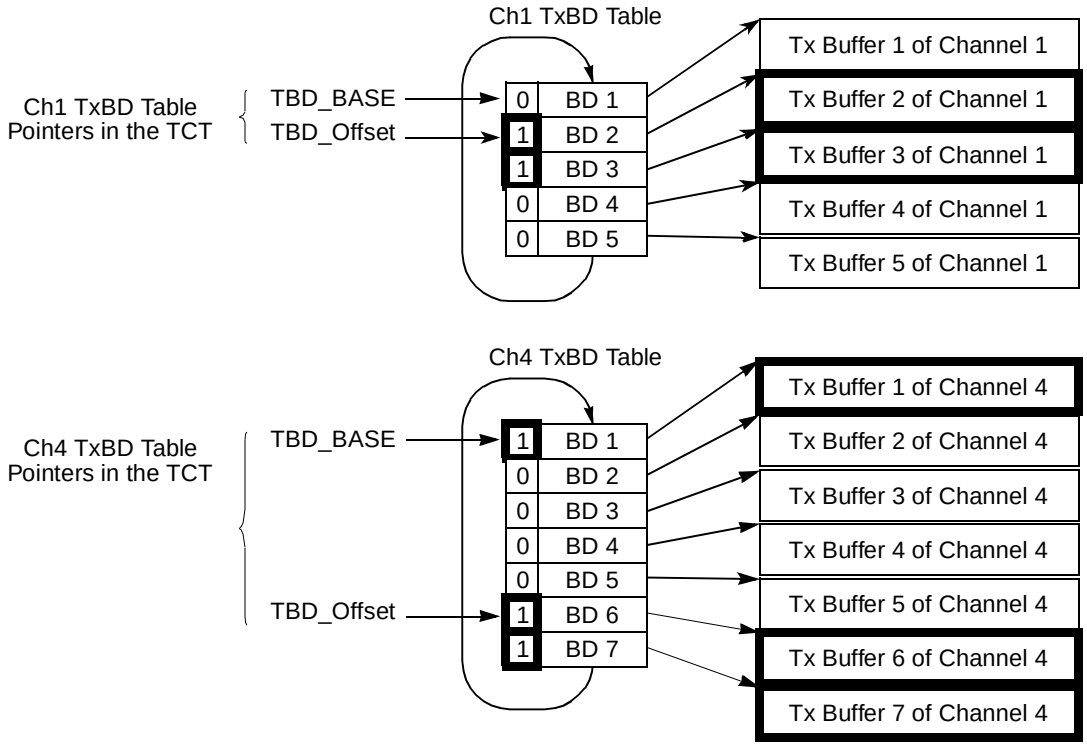
Each ATM channel has separate receive and transmit BD tables. The number of BDs per channel and the size of the buffers is user-defined. The last BD in each table holds a wrap indication. Each BD in the TxBD table points to a buffer to send. At the receive side, the user can choose one of two modes:

- 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 QUICC Engine block 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.

30.3.9.1 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 QUICC Engine block reads the first TxBD in the table and sends its associated buffer. When the current buffer is finished, the QUICC Engine block 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 QUICC Engine block continues sending. If the current BD is not ready, the QUICC Engine block polls the ready bit at the channel rate unless TCT[AVCF] = 1, in which case the QUICC Engine block removes the channel from the APC and clears TCT[VCON]. The core must issue a new ATM TRANSMIT command to restart transmission.

Figure 30-56 shows the ready bit in the TxBD tables and their associated buffers for two example ATM channels.



Note: The highlighted buffers are ready to be sent; unhighlighted buffers are waiting to be prepared.

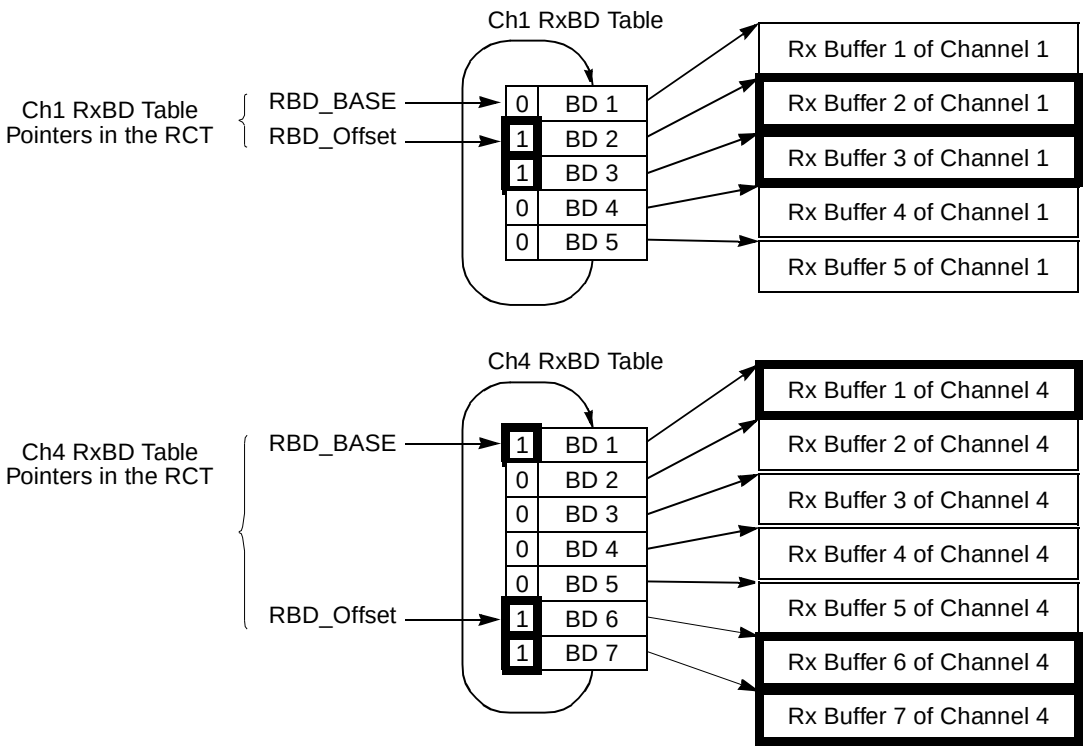
Figure 30-56. Transmit Buffers and BD Table Example

30.3.9.2 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.

30.3.9.2.1 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 QUICC Engine block opens the first BD in the table and starts filling its associated buffer with received data. When the current buffer is full, the QUICC Engine block 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 QUICC Engine block continues receiving. If the BD is not empty, a busy condition has occurred and a busy interrupt is sent to the event queue. Figure 30-57 shows the empty bit in the RxBD tables and their associated buffers for two example ATM channels.



Note: The highlighted buffers are empty; unhighlighted buffers are waiting to be processed.

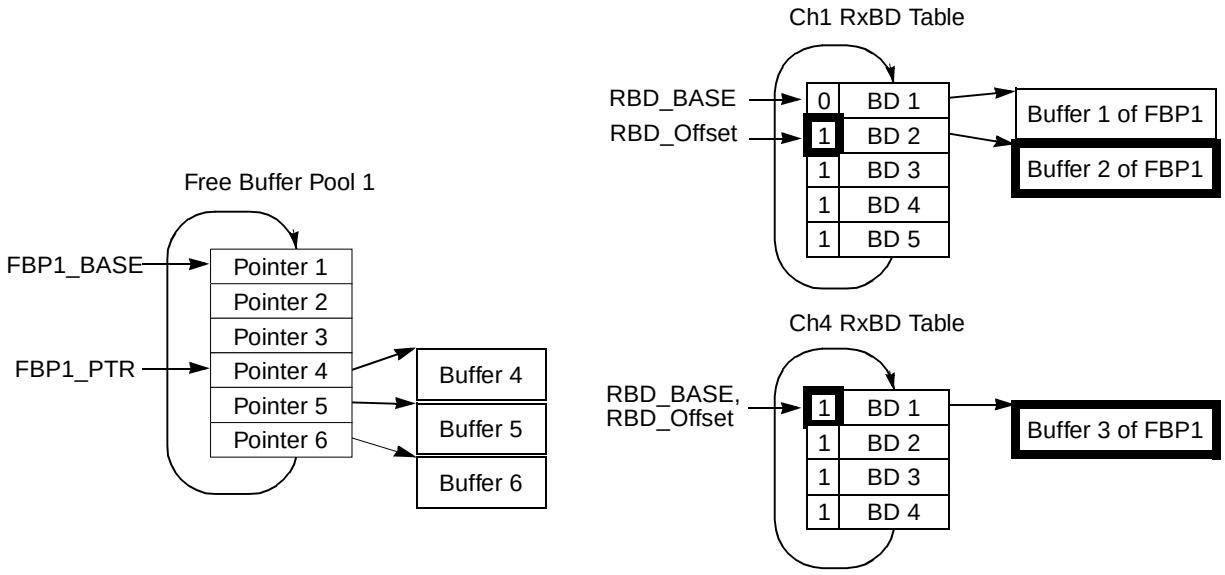
Figure 30-57. Receive Static Buffer Allocation Example

30.3.9.2.2 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 30.3.9.2.3, "Free Buffer Pools."

When an ATM cell arrives, the QUICC Engine block 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 QUICC Engine block 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 QUICC Engine block 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 30-58 shows two ATM channels' BD tables and one free buffer pool. Both channels are associated with free buffer pool 1. The QUICC Engine block allocates the first two buffers of buffer pool 1 to channel 1 and the third to channel 4.



Notes: Buffers 2 and 3 are receiving data. After buffer 1 is processed, it can be returned to the pool.

Figure 30-58. Receive Global Buffer Allocation Example

30.3.9.2.3 Free Buffer Pools

As Figure 30-59 shows, when a buffer pointer is fetched from a pool, the QUICC Engine block clears the entry's valid bit and increments FBP#_PTR. After the QUICC Engine block 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 QUICC Engine block 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 UCCE[GBPB] and a busy interrupt is sent to the interrupt queue specifying the ATM channel code associated with the pool.

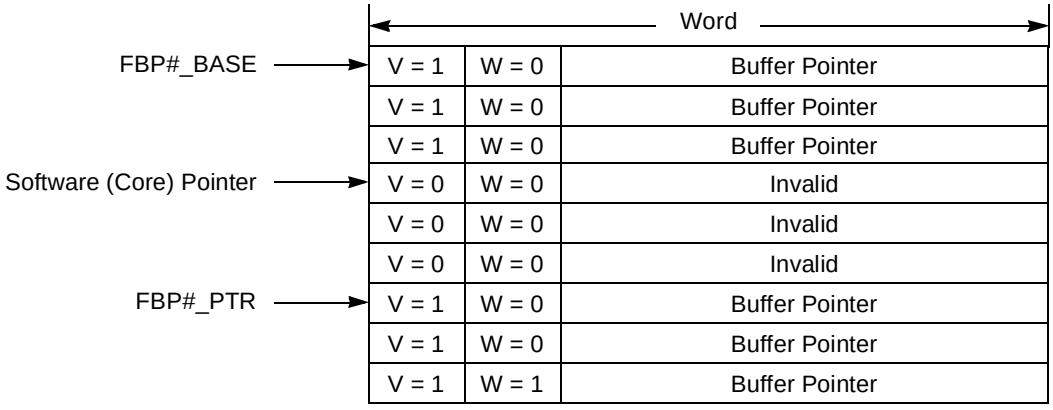


Figure 30-59. Free Buffer Pool Structure

Figure 30-60 describes the structure of a free buffer pool entry.

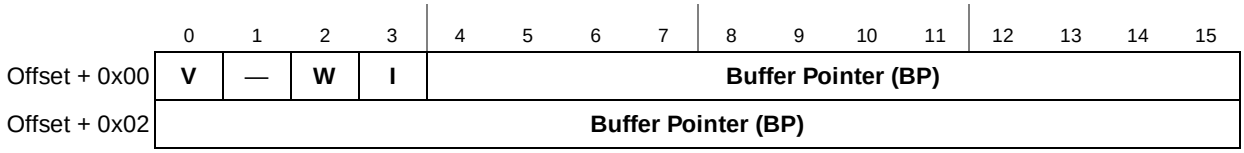


Figure 30-60. Free Buffer Pool Entry

Table 30-51 describes free buffer pool entry fields.

Table 30-51. Free Buffer Pool Entry Field Descriptions

Offset	Bits	Name	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		

30.3.9.2.4 Free Buffer Pool Parameter Tables

The free buffer pool parameters are held in parameter tables in the MURAM. 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.

Management of the FBP is different under different mode of operation and requires different software behavior. In Non Multi-threading mode the operation and programming model is identical to this of the PQII family and is depicted in Table 30-52.

Table 30-52. Free Buffer Pool Parameter Table-Non Multi-Threading Mode

Offset [1]	Bits	Name	Width	Description
0x00	—	**FBP_BASE**	**W**	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**	**W**	Free buffer pool pointer. Pointer to the current entry in the free buffer pool. Initialized to FBP_BASE.
0x08	—	**FBP_ENTRY_EXT**	**HW**	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 30-52. Free Buffer Pool Parameter Table-Non Multi-Threading Mode (continued)

Offset [1]	Bits	Name	Width	Description
0x0A	0	**BUSY**	**HW**	The QUICC Engine block sets this bit when it tries to fetch buffer pointer with V bit clear. UCCE[GBPB] is also set. Initialize to zero.
	1	**RLI**		Red-line interrupt. Set by the QUICC Engine block when it fetches a buffer pointer with I = 1. UCCE[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**	**W**	Free buffer pool entry. Initialize with the first entry of the free buffer pool. Note that FBP_ENTRY must be re initialized 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

The maximum size of the FBP is 16K entries for Multi-thread mode.

Table 30-53. Free Buffer Pool Parameter Table- Multi-Threading Mode

Offset [1]	Bits	Name	Width	Description
0x00	—	**FBP_BASE**	**W**	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_OFFSET_OUT**	**HW**	Free buffer pool QUICC Engine block offset. Offset to the current QUICC Engine block entry in the free buffer pool. **Initialize to 0x04**. The actual address of the FBP entry is FBP_BASE+FBP_OFFSET_OUT. The QUICC Engine block will advance this entry each time a new buffer is taken from the buffer pool. Once reaching the FBP_OFFSET_IN value the QUICC Engine block assumes the pool is empty and no buffer is available
0x06		**FBP_OFFSET_IN**	**HW**	Free buffer pool host offset. Offset to the current entry in the free buffer pool to where the host returns buffer to the pool. Initialize to 0. The actual address of the FBP entry is FBP_BASE+FBP_OFFSET_IN. **Initialize to 0x0.** The Host must maintain this offset in the data structure so that synchronization is kept with the QUICC Engine block. IMPORTANT: The host should never advance this value so it matches the FBP_OFFSET_OUT value. WHen the value of FBP_OFFSET_OUT is equal to the value of FBP_OFFSET_IN it indicates that the pool is empty with no buffers available. In this way the pool size is effectively smaller by one entry.
0x08	—	**FBP_ENTRY_EXT**	**HW**	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 30-53. Free Buffer Pool Parameter Table- Multi-Threading Mode (continued)

Offset [1]	Bits	Name	Width	Description
0x0A	0	**BUSY**	**HW**	The QUICC Engine block sets this bit when it tries to fetch buffer pointer with V bit clear. UCCE[GBPB] is also set. Initialize to zero.
	1	**RLI**		Red-line interrupt. Set by the QUICC Engine block when it fetches a buffer pointer with I = 1. UCCE[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_Size**	**HW**	User initialized to the Size in **bytes** of the FBP entries. (Each entry is 4 bytes.)
0x0E		**—**	**HW**	Reserved- should be cleared

1 Offset from FBT_BASE+RCT[BPOOL] × 16

NOTE

The maximum size of the FBP is 16K entries for Multi-thread mode.

30.3.9.3 ATM Controller Buffers

Table 30-54 describes properties of the ATM receive and transmit buffers for better bus utilization.

Table 30-54. 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
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

30.3.9.4 AAL5 RxBD

Figure 30-61 shows the AAL5 RxBD fields.

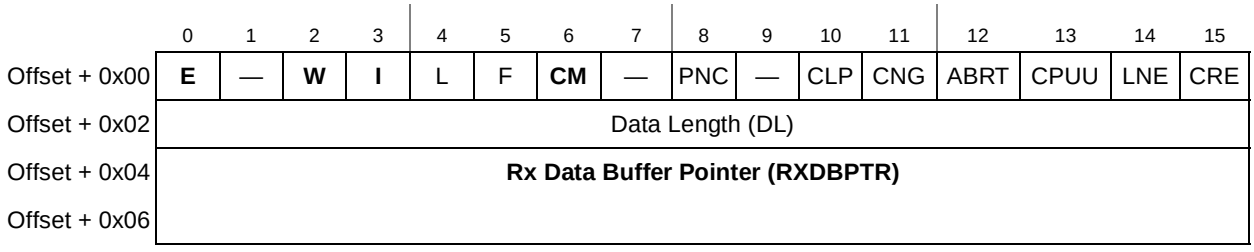


Figure 30-61. AAL5 RxBD

Table 30-55 describes AAL5 RxBD fields.

Table 30-55. AAL5 RxBD Field Descriptions

Offset	Bits	Name	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 QUICC Engine block 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 QUICC Engine block. 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 QUICC Engine block 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. UCCE[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 QUICC Engine block does not clear the empty bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the QUICC Engine block next accesses this BD.
	7	—	Reserved
	8	PNC	UPC Non Conformance. PNC is set by the UPC if the buffer contains a cell, which was not conformed by one of the buckets, but was not dropped. In frame mode (UPCT[UPCM]=10/11), this bit will also be set in the last BD, if the frame has a cell which was not conformed by one (or both) of the buckets, whether it was dropped or not. In GFR mode (UPCT[UPCM]=11), if there is CLP change inside frame or the frame exceed MFS that will also cause setting this bit in the last BD of the frame.
	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.

Table 30-55. AAL5 RxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x00	12	ABRT	Abort message indication. The current message was received with Length field zero. When set the LNE and CRE bits may be set as well and should not be regarded as a different event.
	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. when set the CRE bit could also be set and should not be treated as a different event.
	15	CRE	Rx CRC error. Indicates CRC32 error in the current AAL5 PDU. Set only for the last BD of the frame.
0x02	—	DL	Data length. The number of octets written by the QUICC Engine block into this BD's buffer. It is written by the QUICC Engine block 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. For better bus performance this pointer should be burst-aligned.

Note: In cases where the AAL5 frame length is close to an integer multiple of the MRBLR value specified in the RCT of each ATM channel, it can happen that last BD in a frame which has its L bit set and the DL field value for the complete frame contains no payload data. It contains the AAL5 trailer only and the rest is the AAL5 padding. In this case the BDs which are prior to the last one will have a DL fields which sums up to a number which is equal or greater to the frame length. In this case the application should subtract the difference of frame length from the total sum of the length and ignore this data in the buffer which is prior to the last buffer.

30.3.9.5 AAL0 RxBD

Figure 30-62 shows the AAL0 RxBD.

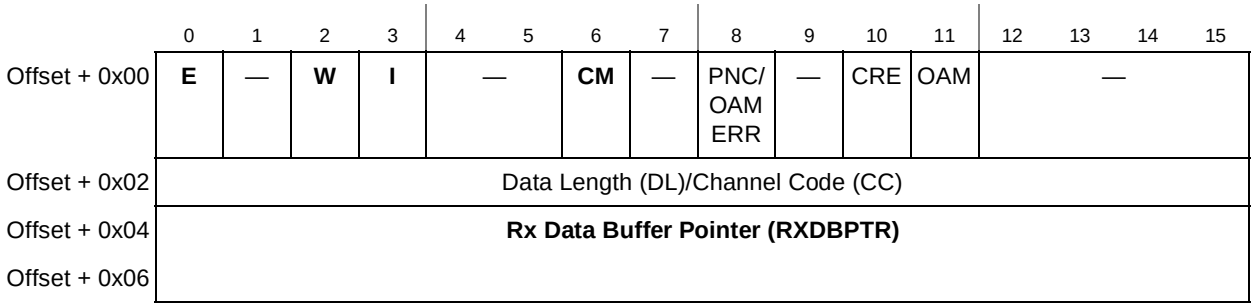


Figure 30-62. AAL0 RxBD

Table 30-56 describes AAL0 RxBD fields.

Table 30-56. AAL0 RxBD Field Descriptions

Offset	Bits	Name	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 QUICC Engine block 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 QUICC Engine block. 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 QUICC Engine block 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. UCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation. 1 The QUICC Engine block does not clear the E bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the QUICC Engine block next accesses this BD.
	7	—	Reserved, should be cleared.
	8	PNC	UPC Non Conformance. PNC - PNC is set by the UPC if the buffer contains a cell, which was not conformed by one (or both) of the buckets, but was not dropped.
0x00	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. If the RCT[TS_EN] is set a four bytes time stamp value is appended to the cell content. In this case the application should allocate 4 more bytes for each buffer.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. For better bus performance this pointer should be burst-aligned.

30.3.9.6 AAL5 User-Defined Cell—RxBD Extension

In user-defined cell mode, the AAL5 RxBDs are extended to 32 bytes; see Figure 30-63.

NOTE

For AAL0, a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

Offset + 0x08	Extra Cell Header. Used to store the user-defined cell's extra cell header. The extra cell header can be 1–12 bytes long.
Offset + 0x14	Reserved (12 bytes)

Figure 30-63. User-Defined Cell—RxBD Extension

30.3.9.7 AAL5 TxBDs

Figure 30-64 shows the AAL5 TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	**R**	—	**W**	**I**	**L**	—	**CM**	—			**CLP**	**CNG**	—			
Offset + 0x02	**Data Length (DL)**															
Offset + 0x04	**Tx Data Buffer Pointer (TXDBPTR)**															
Offset + 0x06																

Figure 30-64. AAL5 TxBD

Table 30-57 describes AAL5 TxBD fields.

Table 30-57. AAL5 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 QUICC Engine block 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 QUICC Engine block 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. UCCE[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 QUICC Engine block does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the QUICC Engine block 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.
0x02	—	**DL**	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the QUICC Engine block. 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 QUICC Engine block.

30.3.9.8 AAL0 TxBDs

Figure 30-65 shows AAL0 TxBDs. Note that the data length field is calculated internally as 52 bytes, plus the extra header length (defined in UPSMR[TEHS]) when in UDC mode.

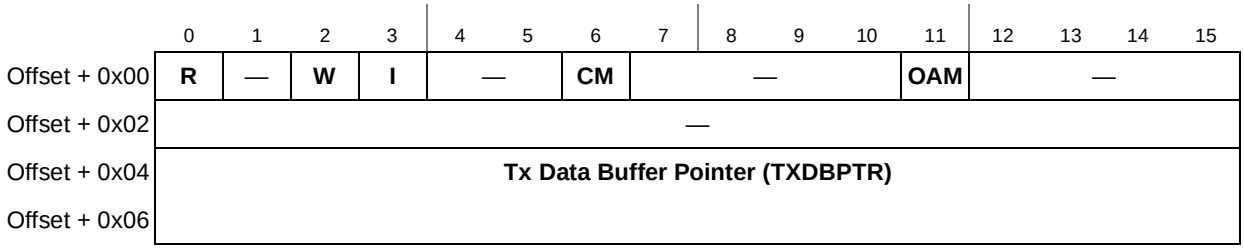


Figure 30-65. AAL0 TxBDs

Table 30-58 describes AAL0 TxBD fields.

Table 30-58. AAL0 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The buffer is not ready for transmission. The user can manipulate this BD or its buffer. The QUICC Engine block 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 QUICC Engine block 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. UCCE[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 QUICC Engine block does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the QUICC Engine block next accesses this BD.
	7–10	—	Reserved, should be cleared.
	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.

Table 30-58. AAL0 TxBD Field Descriptions (continued)

Offset	Bits	Name	Description
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 QUICC Engine block.

30.3.9.9 AAL5 User-Defined Cell—TxBD Extension

In user-defined cell mode, the AAL5 TxBDs are extended to 32 bytes, see Figure 30-66. The extra cell header value should be identical for all the BD's in the ring.

NOTE

For AAL0 a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

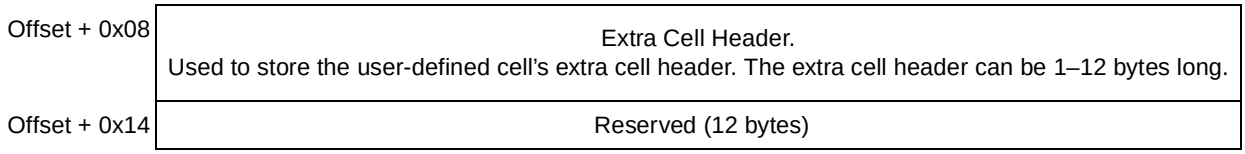


Figure 30-66. User-Defined Cell—TxBD Extension

30.3.10 UPC Structures

30.3.10.1 UPC Table

The UPC Table structure is depicted in Figure 30-67. The field "offset" refers to the address generated in the following way (see PID description in Table 30-5 and Table 30-2):

For internal UPC Tables (PID value 1–31) -

Offset = UPC_PARAM_Table (INT_UPC_BASE) + PID × 48.

For external UPC Tables (PID value 32-1023) -

Offset = UPC_PARAM_Table (EXT_UPC_BASE) + PID × 64.

Note that the UPC table structure is only 48 bytes long. Therefore in external UPC tables, there is some wasted space between the structures. The UPC procedure reads the whole 48 bytes of the UPC table, but writes back only the first 32 bytes, in this way enable software to dynamically change the UPC table parameters which are located in the last 16 bytes.

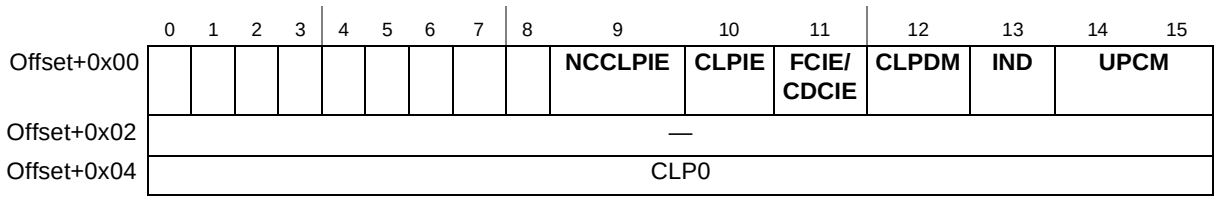


Figure 30-67. UPC Table

Offset+0x06	—		
Offset+0x08	CLP1		
Offset+0x0A	FC		
Offset+ 0x0c	NCCLP0		
Offset+0x0e	NCCLP1		
Offset+0x10	FDC/CDC		
Offset+0x12	B1TF[0:11]]	—	
Offset+0x14	B1TI[0:15]		
Offset+0x16	B1TI[16:31]		
Offset+0x18	**UPCMFS**		
Offset+0x1a	B2TF[0:11]]	—	
Offset+0x1C	B2TI[0:15]		
Offset+0x1E	B2TI[16:31]		
Offset+0x20	**B1II[0:15]**		
Offset+0x22	**B1IF[0:11]**	**DM1**	**LBF1**
Offset+0x24	**B1Lim[0:15]**		
Offset+0x26	**B1Lim[16:31]**		
Offset+0x28	**B2II[0:15]**		
Offset+0x2A	**B2IF[0:11]**	**DM2**	**LBF2**
Offset+0x2C	**B2Lim[0:15]**		
Offset+0x2E	**B2Lim[16:31]**		

Figure 30-67. UPC Table

[1] Bolded parameters are user initialized.

Table 30-59 describes UPC table fields.

Table 30-59. UPC Table Field Descriptions

Offset	Bits	Name	Description
0x00	0–8	—	Reserved, should be cleared.
	9	NCCLPIE	NCCLP0 &NCCLP1 counters overflow interrupt Enable bit. 0 NCCLP0 &NCCLP1 overflow interrupt is disable. 1 NCCLP0 &NCCLP1 overflow interrupt is enable.
	10	CLPIE	CLP0 & CLP1 counters overflow interrupt Enable bit. 0 CLP0 & CLP1 overflow interrupt is disable. 1 CLP0 & CLP1 overflow interrupt is enable.
	11	CDCIE/ FCIE	If UPCM=01 then this bit is Cell Drop Counter overflow interrupt enable bit. if UPCM=10/11 then this bit is Frame Counters (FDC & FC) overflow interrupt enable bit. 0 FDC & FC/CDC overflow interrupt is disable. 1 FDC & FC/CDC overflow interrupt is enable.
	12	CLPDM	CLP Drop Mode. This bit is relevant only in GFR mode [UPCM=11]). Should be cleared otherwise. 0 Do not drop cells due to CLP value. 1 Drop remain cells of frame if there is CLP change within the frame.
	13	IND	Independent bit. 0 If a cell is dropped by the UPC, the theoretical arrival time, of both buckets will NOT be updated. **Important**: When working in this mode VP policing is not allowed 1 The bucket theoretical arrival time is updated only due to the bucket decision.
	14–15	UPCM	UPC operation Mode. 00 Not defined. Should not be used. 01 Cell base mode. See 30.2.20.2.1/30-40. 10 Frame awareness mode. See 30.2.20.2.2/30-40 (applied only for AAL5). 11 GFR mode. See 30.2.20.2.3/30-40. The UPC work as frame awareness mode, and also MFS and CLP changes are checked (applied only in AAL5).
0x02	—	—	Reserved, should be cleared.
0x04	—	CLP0	16 bit counter for all cells with CLP=0. Maskable interrupt is asserted upon wrap of counter. Initialize to zero.
0x06	—		Reserved, should be cleared.
0x08	—	CLP1	16 bit counter for all cells with CLP=1. Maskable interrupt is asserted upon wrap of counter. Initialize to zero.
0x0a	—	FC	Frame Counter. Count all the frame which go through the policer. Maskable interrupt is asserted upon wrap of counter. Initialize to zero.
0x0c	—	NCCLP0	16 bit counter for CLP0 cells which were not conformed by both buckets. Maskable interrupt is asserted upon wrap of counter. Initialize to zero.
0x0e	—	NCCLP1	16 bit counter for CLP1 cells which were not conformed by both buckets. Maskable interrupt is asserted upon wrap of counter. Initialize to zero.
0x10	—	FDC/CDC	Frames Drop Counter/Cell Drop Counter. 16 bit counter. If UPCM=10/11, When frame or part of it is dropped by the UPC, FDC is updated. If UPCM=01, When cell is dropped by the UPC, CDC is updated. Maskable interrupt is asserted upon wrap of counter. Initialize to zero.

Table 30-59. UPC Table Field Descriptions (continued)

Offset	Bits	Name	Description
0x12	0–11	B1TF[0–11]	Bucket1 Theoretical arrival time, Fraction part.
	12–15		Reserved, should be cleared
0x14	—	B1TI[0–31]	Bucket1 Theoretical arrival time, Integer part.
0x18	—	UPCMFS	UPC Maximum Frame Size. This field should be programed to (MFS-1)*48, where MFS is maximum frame size in cell unit. This field is relevant only in GFR mode [UPCMD=11]). Should be cleared otherwise.
0x1a	0–11	B2TF[0–11]	Bucket2 Theoretical arrival time, Fraction part.
	12–15		Reserved, should be cleared.
0x1c	—	B2TI[0–31]	Bucket2 Theoretical arrival time, Integer part.
0x20	—	B1II	Bucket1 Increment Integer part.
0x22	0–11	B1IF[0–11]	Bucket1 Increment Fraction part.
	12	DM1	Bucket1 nonconforming Discard Mode. 0 Cell which was not conformed by bucket1 will be tagged. 1 Cell which was not conformed by bucket1 will be dropped. Tagging is done by setting PNC bit in RXBD. In frame mode (UPCM=10/11), if a cell is dropped, all remaining cells of that frame are dropped, except the last cell (in this case the PNC bit in the last RXBD will be set). If the first cell of a frame is dropped, then the whole frame is dropped.
	13–15	LBF1	Leaky Bucket1 Filter, see 0xLBF - The Leaky Bucket Filter
0x24	—	B1Lim[0–31]	Bucket1 Limit. Most Significant bit must be cleared.
0x28	—	B2II	Bucket2 Increment Integer part.
0x2a	0–11	B2IF[0–11]	Bucket2 Increment Fraction part.
	12	DM2	Bucket2 nonconforming Discard Mode. 0 Cell which was not conformed by bucket2 will be tagged. 1 Cell which was not conformed by bucket2 will be dropped. Tagging is done by setting PNC bit in RXBD. In frame mode (UPCM=10/11), if a cell is dropped, all remaining cells of that frame are dropped, except the last cell (in this case the PNC bit in the last RXBD will be set). If the first cell of a frame is dropped, then the whole frame is dropped.
	13–15	LBF2	Leaky Bucket2 Filter, see 0xLBF - The Leaky Bucket Filter
0x2c	—	B2Lim[0–31]	Bucket2 Limit. Most Significant bit must be cleared.

30.3.11 UNI Statistics Table

The UNI statistics table, shown in Table 30-60, resides in the multi-user 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# × 4.

Table 30-60. 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 (for example, is late or is missing).
0x02	MIC_COUNT	Hword	Counts miss-inserted cells dropped as a result of address look-up failure.

1 Offset from UNI_STATT_BASE+PHY# × 4

30.3.12 ATM Exceptions

The ATM controller interrupt handling involves two principal data structures: UCCEs (UCC event registers) and circular interrupt queues.

Five priority interrupt queues are available.

Four queues are available for termination mode 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. These queues are numbered 0–3.

After an interrupt request, the host reads UCCE. If UCCE[GINT*x*] = 1, at least one entry was added to one of the interrupt queues. After clearing UCCE[GINT*x*], the host processes all 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 30.3.12.3, "ATM Termination Interrupt Queue Entry." This procedure implies that it is possible to get an interrupt and that the interrupt queue doesn't contain any valid entry as it was already processed in the previous interrupt handling.

The host controls the number of interrupts sent to the core using a down counter in the interrupt queue's parameter table. 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, UCCE[GINT*x*], is set. In Multi Threading operation this counter is indicating a minimum value for generating an interrupt.

30.3.12.1 ATM Event Register (UCCE)/Mask Register (UCCM)

The UCCE register is the ATM controller event register when the UCC operates in ATM mode. When it recognizes an event, the ATM controller sets the corresponding UCCE bit. Interrupts generated by this register can be masked in UCCM. UCCE 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 QUICC Engine block clears the internal interrupt request.

UCCM is the ATM controller mask register. The UCCM has the same bit format as UCCE. Setting an UCCM bit enables and clearing a bit masks the corresponding interrupt in the UCCE.

Figure 30-68 shows UCCE and UCCM bits.

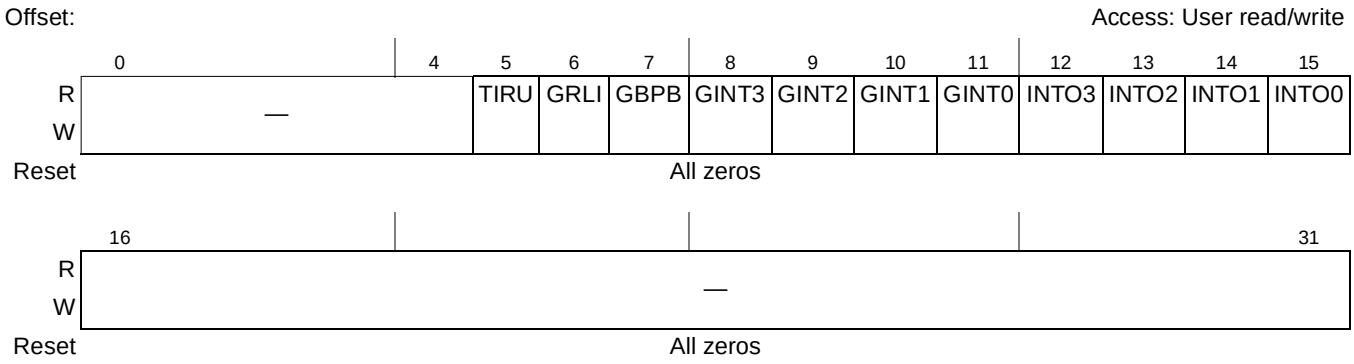


Figure 30-68. UCC Event Register (UCCE)/Mask Register (UCCM)

Table 30-61 describes UCCE fields.

Table 30-61. UCCE/UCCM Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	TIRU	Transmit internal rate underrun. A transmit internal rate counter expired and a cell was not sent because the transmit FIFO was empty. TIRU may be set only when using transmit internal rate mode.
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 30.3.12, "ATM Exceptions."
12–15	INTO*x*	Interrupt queue overflow. Set when an overflow condition occurs in the corresponding interrupt queue. This occurs when the QUICC Engine block attempts to overwrite a valid interrupt entry. See Section 30.3.12.2, "Interrupt Queues."
16–31	—	Reserved, should be cleared.

30.3.12.2 Interrupt Queues

Interrupt queues are located in external memory. The parameters of each queue are stored in a table. See Table 30-63 and Table 30-64

When an interrupt occurs, the QUICC Engine block writes a new entry to the interrupt queue, the V bit is set, and the queue pointer (INTQ_PTR) is incremented. Once the QUICC Engine block uses an entry with W = 1, it returns to the first entry in the queue. If the QUICC Engine block tries to overwrite a valid entry (V = 1), an overflow condition occurs and the queue's overflow flag, UCCE[INTO*x*], is set.

The interrupt queue structure is displayed in Figure 30-69.

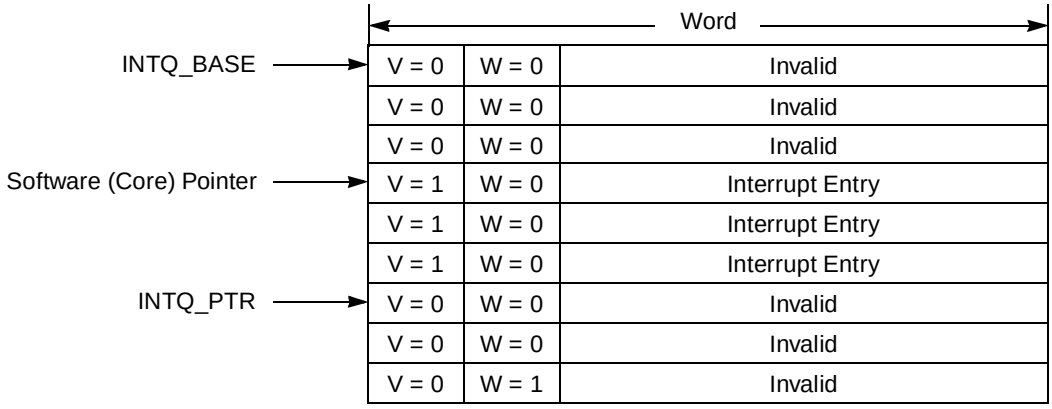


Figure 30-69. Interrupt Queue Structure

30.3.12.3 ATM Termination Interrupt Queue Entry

Each one-word interrupt queue entry provides detailed interrupt information to the host. Figure 30-70 shows an entry for Non UPC- Policer events

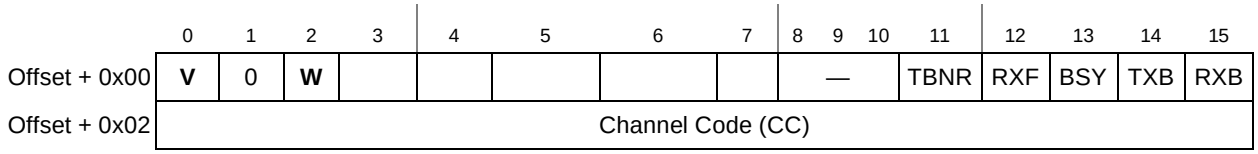


Figure 30-70. Interrupt Queue Entry—Non UPC

Figure 30-71 shows an entry which is a result of a UPC-ATM Policer event. Bits 1,10 and 12 are set in order to distinguish between a UPC event and all other events on the ATM.

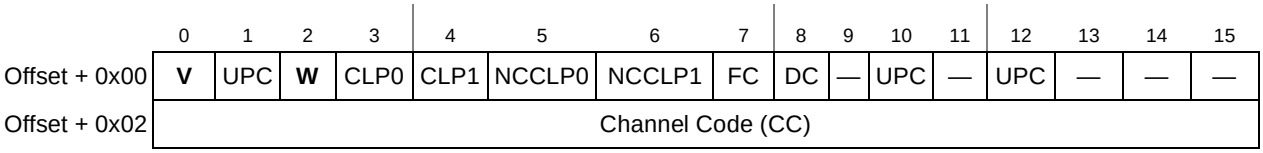


Figure 30-71. Interrupt Queue Entry—UPC Event

30.3.12.4 Interrupt Queue Parameter Table

Table 30-62 describes interrupt queue entry fields.

Table 30-62. Interrupt Queue Entry Field Description

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be use by the QUICC Engine block. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	UPC	UPC interrupt bit. 0 This interrupt was not generated by the UPC. 1 This interrupt was generated by the UPC.

Table 30-62. Interrupt Queue Entry Field Description (continued)

Offset	Bits	Name	Description
0x00	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	CLP0	CLP0 counter overflow. Valid only if UPC bit is set. 0 No CLP0 counter overflow. 1 CLP0 counter overflow.
	4	CLP1	CLP1 counter overflow.Valid only if UPC bit is set. 0 No CLP1 counter overflow. 1 CLP1 counter overflow.
	5	NCCLP0	NCCLP0 counter overflow.Valid only if UPC bit is set. 0 No NCCLP0 counter overflow. 1 NCCLP0 counter overflow.
	6	NCCLP1	NCCLP1 counter overflow.Valid only if UPC bit is set. 0 No NCCLP1 counter overflow. 1 NCCLP1 counter overflow.
	7	FC	Frame Counter overflow.Valid only if UPC bit is set. 0 No FC counter overflow. 1 FC overflow.
	8	DC	Cell/Frame Discard Counter.Valid only if UPC bit is set. 0 No FDC/CDC overflow. 1 FDC/CDC overflow.
	9–10	—	Reserved.
	11	TBNR	Tx buffer-not-ready. Set when a transmit buffer-not-ready interrupt is issued. This interrupt is issued when the QUICC Engine block 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.

The interrupt queue parameters are held in parameter tables in the multi-user RAM; This table has a different programming model when the QUICC Engine block operates in a backwards compatible mode and when Multi-Threading mode is activated. Table 30-63 and Table 30-64 describes the parameter tables programming under each mode of operation. 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.

Important:

- For proper operation Receiver and transmitter interrupts should reside on different interrupt queues
- Each UCC should assign it own interrupt queue management tables. They should not be shared by several UCCs.

Table 30-63. PQII-like Interrupt Queue Parameter Table-Non Multi-Threading

Offset [1]	Name	Width	Description
0x00	**INTQ_BASE**	Word	Base address of the interrupt queue. User-defined.
0x04	**INTQ_PTR**	Word	QUICC Engine block actual Pointer to interrupt queue entry. Initialize to INTQ_BASE. Managed by the QUICC Engine block.
0x08	**INT_CNT**	Half Word	Interrupt counter. Initialize with INT_ICNT. The QUICC Engine block decrements INT_CNT for each interrupt. When INT_CNT reaches zero, the queue's global interrupt flag UCCE[GINT*x*] is set.
0x0A	**INT_ICNT**	Half Word	Interrupt initial count. User-defined global interrupt threshold—the number of interrupts required before the QUICC Engine block issues a global interrupt (UCCE[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 reenable interrupt processing.

Note:

1 Offset from INTT_BASE+RCT/TCT[INTQ] × 16

NOTE

When working in Non Multi-Threading mode the events of the receiver and transmitter should reside on different interrupt queues.

Table 30-64. Multi-Threading Mode Interrupt Queue Parameter Table

Offset [1]	Name	Width	Description
0x00	**INTQ_BASE**	Word	Base address of the interrupt queue. User-defined.
0x04	**INTQ_OFFSET_OUT**	Half Word	The software offset to interrupt queue entry. Initialize to 0. Managed by the software (host). The actual host entry pointer is equal to INTQ_BASE+INTQ_OFFSET_OUT.
0x06	**INTQ_OFFSET IN**	Half Word	The QUICC Engine offset to the interrupt queue entry. Initialize to 0. Managed by the QUICC Engine block. The actual QUICC Engine entry pointer is equal to INTQ_BASE+INTQ_OFFSET_IN.
0x08	**INT_CNT**	Half Word	Interrupt counter. Initialize with INT_ICNT. The QUICC Engine block decrements INT_CNT for each interrupt. When INT_CNT reaches zero, the queue's global interrupt flag UCCE[GINT*x*] is set.
0x0A	**INT_ICNT**	Half Word	Interrupt initial count. User-defined global interrupt threshold—the number of interrupts required before the QUICC Engine block issues a global interrupt (UCCE[GINT*x*]).
0x0C	**—**	Half Word	Reserved. Should be cleared

Table 30-64. Multi-Threading Mode Interrupt Queue Parameter Table (continued)

Offset [1]	Name	Width	Description
0x0E	**INTQ_Size**	Half Word	Size in bytes of the interrupt queue entries table.

Note:

1 Offset from INTT_BASE+RCT/TCT[INTQ] × 16

NOTE

For Multi-thread mode the maximum size of the interrupt queue is limited to 16 K entries.

30.4 ATM Controller—Application Information

30.4.1 ATM Commands

The host initializes and activates the ATM operation using the host commands.

This is done by two memory mapped registers depicted here: the CECE and the CECDR. Detailed description is located at RISC control chapter:

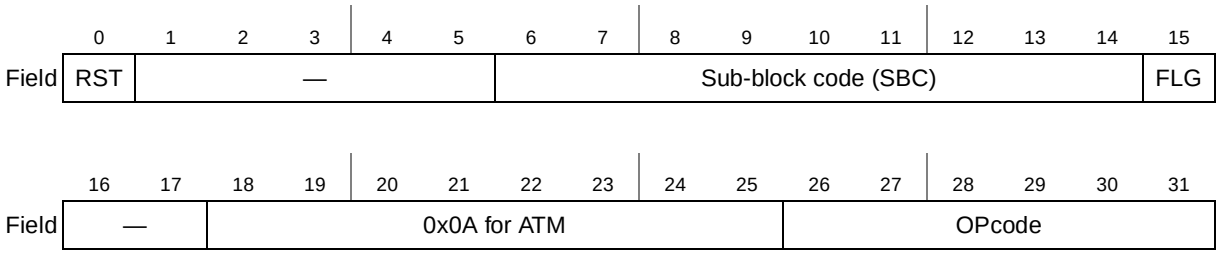


Figure 30-72. CE Command Register (CECR)

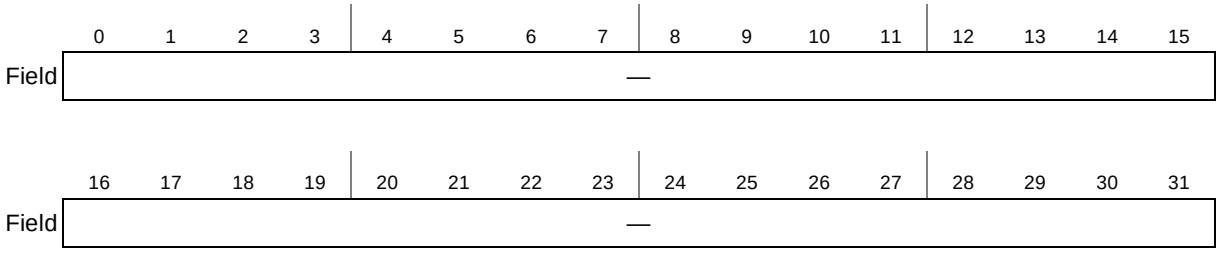


Figure 30-73. CE Command Data Register (CECDR)

The host commands for the ATM are explained in the following sections.

30.4.1.1 ATM Transmit Command

The ATM transmit command 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. The CECR register has the following settings:

Opcode is:0x001010

SBC: determined by the UCC.

Before issuing the command, the user should initialize COMM_INFO fields in the Global ATM Parameters described in Table 30-19 The information is described in Figure 30-74.

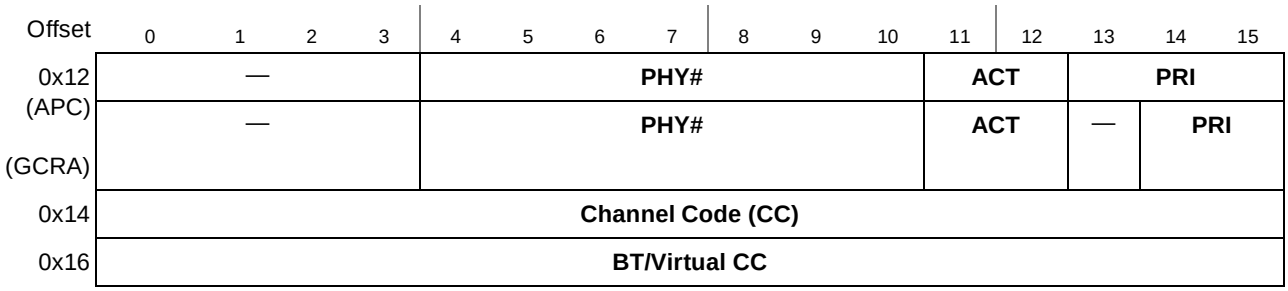


Figure 30-74. COMM_INFO Field

NOTE
Backward Compatibility

The PHY# and the CTB bits locations were changed.

Table 30-65 describes COMM_INFO fields.

Table 30-65. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x12	0	—	Reserved, should be cleared.
	1–2	—	Reserved, should be cleared.
	3	—	Reserved, should be cleared.
	4–10	PHY#	**Note:** 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 10 GBR-UBR channel. This is valid for inserting the lower priority UBR channel for transmission. 11 For Hierarchical frame based activation of a channel into the WFQ selection
	13–15	PRI (APC)	APC priority level. 000 Highest priority (APC_LEVEL1) 111 Lowest priority (APC_LEVEL8).
	13	—(GCRA)	Reserved, should be cleared.
	14–15	PRI (GCRA)	For GCRA scheduler only 4 priority levels are available. 0x00–0x03

Table 30-65. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x14	0–15	CC	Channel code. The channel code associated with the current channel.
0x16	0–15	BT/ Virtual CC	Burst tolerance. For use by VBR channels only (ACT field is 0b01). Specifies the initial burst tolerance (GCRA burst credit) of the current VC. When ACT=0b11, for example, inserting a channel into the hierarchical frame based scheduling this entry is the CC of the Virtual channel acting as the root of the hierarchy.

30.4.1.2 Assign Page

The host can assign each UCC its page and override the default page assignment for the UCCs.

This is done by issuing the Assign Page command.

When using this command the host should first issue the command for the desired UCC. Then initialize all the parameters in the allocated page and only then continue with the Multi-threading and terminator initialization.

30.4.1.3 ATM Multi-Thread Init

This command will initialize the UCC for a multi-threading operation. It should be issued only once for all the UCCs which are sharing the same thread pool.

The command parameters are:

Opcode: 0x010000

SBC should match the snum of one of the UCCs who are utilizing the common multi-threading table. The information in Table 30-20 and Table 30-30 should be initialized by the host prior to issuing this command.

The QUICC Engine block will scan the Multi-threading table and assign each of the threads the MURAM allocated to it by the host.

30.4.2 Configuring the ATM Controller for Maximum QUICC Engine Performance

The following sections recommend ATM controller configurations to maximize QUICC Engine performance.

30.4.2.1 Configuration of Internal Rate Timers

See Section 31.3.2, “Transmit Internal/External Rate Modes.”

30.4.2.2 APC Configuration

30.4.2.2.1 APC CPS

Maximizing the number of cells per slot (CPS) defined in the APC data structure improves QUICC Engine performance. CPS defines the maximum number of ATM cells allowed to be sent during a time slot. (See Section 30.2.5.3.1, “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.

30.4.2.2.2 APC Priority Levels

Minimizing the number of priority levels defined in the APC data structure improves QUICC Engine performance. The user can configure the APC data structure to have from one to eight priority levels. (See Section 30.2.11, “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.

30.4.2.2.3 APC Flux Compensation

For variable bit rate PHY (for example, radio link), which can fall below the transmission bandwidth defined by the APC internal rate timers, the ATM APC does not use the remaining bandwidth for higher priority traffic only. Using the APC Flux Compensation mechanism could improve the performance of this system. Also it could be useful for supporting system with many UBR connections, see Section 30.2.8, “APC Flux Compensation”.

The penalty for using the APC Flux Compensation mechanism is in terms of QUICC Engine performance.

30.4.2.2.4 Scalable APC mode

If an ATM system requires connections with very high variance in bit rates that spans orders of magnitude, the code offers a way to reduce the space required by the APC scheduling table without impacting the scheduling itself. Also it is very useful if we have Multi-PHY application (up to 127 PHYs), and we would like to reduce the APC table sizes in the multi-user RAM as much as possible.

The penalty for using the Scalable APC mode is:

1. Reducing the QUICC Engine performance.
2. Increase the cell delay variation time.

30.4.2.2.5 UBR+ prioritized mode

Systems that requires several priority levels for UBR+ channels, the prioritized UBR+ mechanism should be enabled. The penalty for using this mode is in terms of QUICC Engine performance.

30.4.2.3 GCRA Scheduler mode

As described in Section 30.2.12, “GCRA Scheduler”, the user should use the GCRA scheduler mode for systems where the number of PHYs per UCC is big and the number of channels per PHY is relative small

(up to 64 channels), and the variance between the channels rate is big. This is normally the case in DSLAM. The big advantage of the GCRA scheduler method compared with the traditional APC algorithm, is the economical RAM usage, and the improvement of performance.

30.4.2.4 ATM Multi-thread Mode

As described in Section 30.2.3, “ATM Multi-Threading”, in order to support higher ATM traffic rates the user should use the Multi-Threading mode. This mode enables better bus/QUICC Engine block utilization, by using additional available request resources in the QUICC Engine block. This mode is enabled per UCC by configuring in the CECR[MCN], when issuing an UCC init Rx/Tx parameters command.

30.4.2.5 Buffer Configuration

Using statically allocated buffers of optimal sizes also improves QUICC Engine performance:

- Buffer size. Opening and closing buffer descriptors consumes QUICC Engine block processing time. Because smaller buffers require more opening and closing of BDs, the optimal buffer size for maximum QUICC Engine performance is equal to the packet size (an AAL5 frame, for example).
- Free buffer pool. When the free buffer pool is used, the QUICC Engine block 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 30.3.9.2, “Receive Buffer Operation.”) When allowed by the application, use static buffer allocation to increase QUICC Engine performance.

Chapter 31
UTOPIA L2 Bus Controller (UPC)

31.1 Overview

NOTE

The MPC8323 supports only one UTOPIA interface (8-bit data on UPC1), via three UCCs (UCC1, UCC3 and UCC5), a single device (Device 1) and up to 31 PHYs.

The UTOPIA L2 bus Controller (UPC) is the UTOPIA MAC peripheral of the QUICC Engine 1.0 block. It supports UTOPIA level 2 for both master and slave modes. The UPC is a MAC, and therefore is not independent from the UCCs, which can be roughly described as the FIFOs or queues. Unlike the other MACs of the QUICC Engine 1.0 block, which are integrated within the UCC, the UPC can route the data path to one of three UCCs[1] (see Figure 31-1 and the routing example in Figure 31-2).

As master, the UPC controls one UL2 bus with up to 31 PHYs. The group of PHYs controlled by a common Clav/Enb signals pair are called a device within this chapter. Each device can be either a multiple PHY or a single PHY. As slave, the UPC can operate in SPHY or in MPHY mode. In slave mode configuration only a single UCC is routed to the UPC.

Transmit and receive functions on the UPC are independent, and can be set to either master or slave mode. The device routing is also configured independently for Rx and Tx. As transmitter, the UPC supports an internal rate mechanism for allocating a precise bandwidth to each port in an MPHY or SPHY distribution. The transmission rate is determined by the UPC internal rate timers. The source clock of the internal rate timers can be the serial clock, or it can be derived from a baud-rate generator or from an external clock source. The internal rate works differently for SPHY and MPHY devices. In SPHY mode, the internal rate paces the line rate, and the transmit FIFO becomes a leaky bucket. In MPHY mode, the internal rate paces the FIFO fill rate; that is, the polling status is qualified by the internal rate, and the FIFO only contains cells for PHYs that are actively requesting data. In the receive direction, a selection based on programmable priority is used to prioritize cell transfers from the ports to the UCC.

1. UPC1 can be routed only to UCC1, UCC3, UCC5.

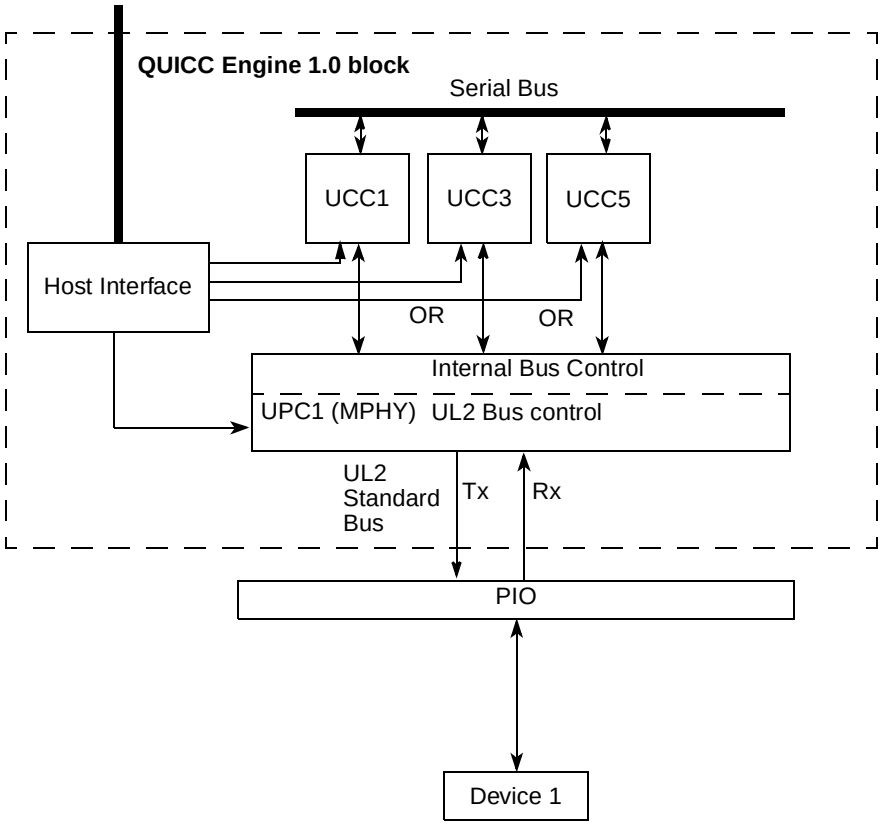


Figure 31-1. QUICC Engine 1.0 Block with UPC1 (1 Device, up to 31 Ports)

31.2 Features

31.2.1 UPC Features

- Connection to 1 device through one UL2 Standard bus I/F
- Up to 31 addresses for 1 physical device in normal address mode (5-bit address)
- Connection to one of 3 UCCs, with deep, programmable size FIFOs
- Slave can release its outputs to high-impedance when not selected (Slave in MPHY system).
- Internal loopback mode

NOTE

To maximize performance, the UTOPIA bus bandwidth should be at least 1.1 times the aggregated throughput on the bus.

31.2.2 UL2 Features

- Conforms to ATF UL2 standard version 1.0, June 1995
- 8 bit UL2 bus
- Cell level handshake support
- Single device (device 1), using xClav/xEnb pair index 0.
- SPHY: Device can be configured as SPHY. Clav is assumed as direct status or as always valid.
- MPHY: Device can be configured as MPHY, with single Clav polling method.
- Default address mode (5-bit address) for 31 PHYs.
- Internal rate:
  - 4 internal rates
  - Programmable max credit value (Useful for PON or xDSL applications).
- Tx internally discards idle cells.
- Rx discard idle cells option.
- Tx scheduling:
  - SPHY: Internal rate shapes the actual transmission rate on the UTOPIA bus. Cells are prefetched to the virtual TxFIFO.
  - Request arbitration is handled in a fixed or round robin (for fairness) priority scheme among pending qualified PHYs.
  - SPHY in multi-port mode: up to 32 logical ports with internal rate and fixed or round priority among them are funneled to the same PHY.
- Tx cell transfer
  - Back to back cell transfer without dead cycle in SPHY.
  - Dead cycle on back to back transfer on MPHY or when changing device.
  - Programmable HEC field.
  - Automatic data parity generation.
- Rx selection (two priority levels decision):
  - 2 priority levels per PHY (reduce worst case round trip latency time for high priority PHY).
  - Arbitration is in round robin priority among all PHYs within a device of the same priority level.
- Rx cell transfer
  - Optional idle cell discard
  - Back to back cell transfer with single dead cycle.
  - Optional HEC check (COSET optional)
  - Optional data parity check (Odd parity)
  - Framing error detection (missing SOC error)
  - Error on RSOC violation
- Polling
  - Polling is periodic and fair.

  - Independently programmable last PHY address for Rx and for Tx.
  - Maskable polling per port (Rx, Tx or both).
- Slave mode
  - Slave is using device 1 xClav/xEnb control pairs.
  - Direct status polling method in SPHY mode.
  - Configurable slave address in MPHY mode (independent Rx and Tx settings).
  - Support pause, halt (master initiated TxEnb, RxEnb flow control).
  - Support back to back Tx cell transfer (without negation of TxEnb).

31.2.3 Internal Rate Features

Internal rate paces the transmit FIFO dequeuing rate for SPHY and the enqueuing rate for MPHY. In general, the internal rate is set in the initialization process, but it can be changed dynamically. Each PHY has an expiration counter that tracks transmit rate variation (jitter) in the transmit rate, which can be compensated by a burst of cells at a rate higher than the prescribed internal rate. If this variation exceeds a programmable watermark, a transmit time-out event is reported.

- Support external rate (internal rate timers disabled)
- Fastest PHY to slowest PHY transmit rate ratio of 32768 (for example, the fastest PHY can be set to 155 Mbps and the slowest PHY to 5 Kbps).
- Configurable expiration counter threshold (cell/packet burst size, 1 to 16). When setting to 1 (Cell per second mode), transmit internal rate underrun is not reported.
- Sub rate dividers:
  - Serial clock as the basic rate for internal sub-rates. (An internal BRG clock could also be used).
  - Pre-scalar divider for each device.
  - 4 sub rates. Each PHY in a device can be assigned any of those rates.
  - Divide clock up to 32768 of the basic rate.
- Transmit internal rate underrun event per PHY, in the event of a mismatch of up to 16 cells/segments[1] over any interval of time between the number of PHY transmit requests (which are polled at the Internal Rate) and the actual number of transmitted cells/packets.

31.3 Modes of Operation

31.3.1 UTOPIA Mode

The QUICC Engine 1.0 ATM controller interfaces with a PHY device through the UPC UTOPIA interface. The QUICC Engine 1.0 block supports UTOPIA level 2 for both master and slave modes.

1. 16 or less, configured in UPRP[TIREC].

31.3.1.1 UTOPIA Master Mode

As master, the UPC controls one UL2 bus with one device and can route it to any one of three UCCs. Each device can operate as an SPHY or MPHY.

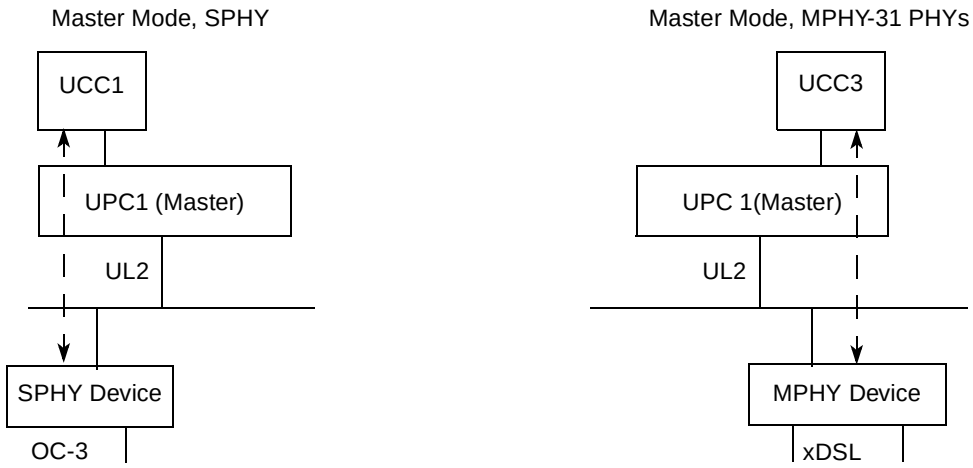


Figure 31-2. Example: Device Configurations in Master Mode

31.3.1.1.1 UTOPIA Master MPHY Operation

The cell transfer in a multiple PHY ATM port uses cell-level handshaking as defined in the UTOPIA standards. When a device is configured to operate as a MPHY, the UPC as a transmitter fetches a cell for any port in the MPHY device if there is a pending, qualified Clav for that port. During the polling phase, the PHYs sample the address bus and the device uses its Clav to indicate the status of the corresponding PHY. In the selection phase, the selected PHY address appears on the common address bus and at the following cycle the Enb signal of the selected device is asserted.

31.3.1.1.2 Addressing in MPHY Mode

Up to 31 PHYs can be supported on a device. The routing method for default MPHY mode devices is displayed in Figure 31-3.

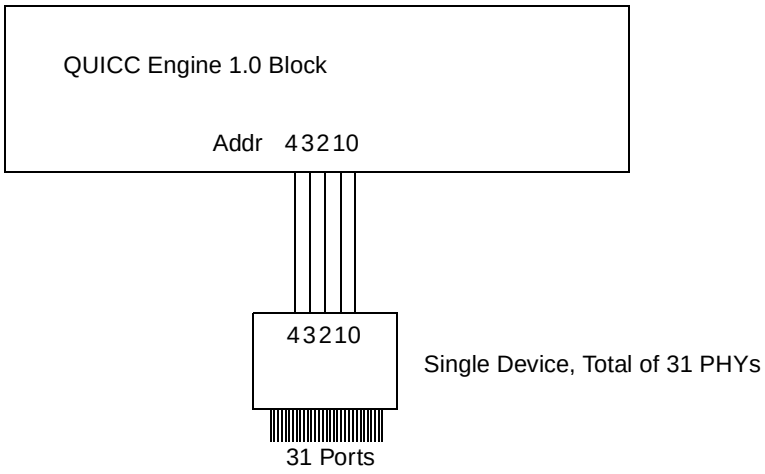


Figure 31-3. MPHY Devices Address Bus Routing

31.3.1.2 UTOPIA Slave Mode

The UPC can function as a single UL2 slave in SPHY or MPHY mode. The slave is always configured as device 1 (registers, I/O) and it can be routed to one of any of the three UCCs. The UPC uses direct status polling for SPHY and clav status polling for PHYs in the MPHY system. By default, the slave is represented internally as port 0. If UPDC[PE] = 0, the port enable register has no function in slave mode. If UPDC[PE] != 0, PPER[0] enables the slave regardless of the PHY address. For a slave in a multi-endpoint configuration, PPER[0:31] is used in the same way as for a master.

31.3.1.2.1 UTOPIA Slave MPHY Operation

The QUICC Engine block supports one slave in a MPHY configuration. The slave address is configurable, with independent addresses for the Tx slave and the Rx slave. For a UTOPIA slave in MPHY system mode, cells are transferred using a cell-level handshake as defined by the UTOPIA level-2 standard. The user should write the ATM controller PHY address in UPLPAy[PHY ID], enable the corresponding ports in UPERy, and configure the ATM controller PHY-number dependent structures accordingly (APC, Lookup tables, and so forth).

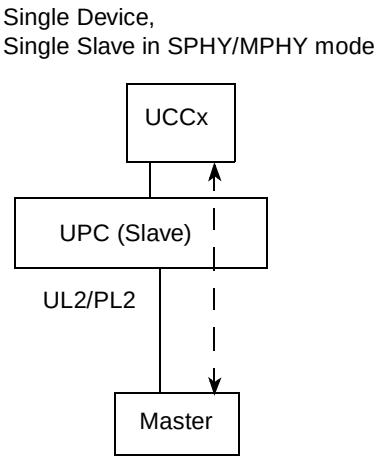


Figure 31-4. SPHY/MPHY Configuration in Slave Mode

31.3.2 Transmit Internal/External Rate Modes

The UPC controller supports the following two rate modes (controlled by the UPRP*x* register):

- **External rate (ER) mode** (UPRPx[TIREC] = 0)—The total transmission rate is determined by the PHY transmission rate. In ATM protocols, Idle cell transmission is controlled by UCC_Modes soft register. Idle cell transmission should be enabled to achieve accurate scheduling.
- **Internal rate (IR) mode—**The total transmission rate is determined by the UPC internal rate timers. The internal rate mechanism is supported for up to 32 PHYs on 4 devices. Each device has its own 4 internal rate values, defined in UPTIRR*x*. The UPTIRR*x* includes the initial value of the internal rate timer. Transmit of a cell/packet is enabled when an internal rate timer expires.

31.3.2.1 Using Transmit Internal Rate Mode

Internal rate programming sequence:

1. Calculate the cell to cell time delay (Delay): Delay= (Number of bits in cell or packet) / (Required bit rate in Mbps).
2. Calculate the cell to cell bus cycles delay (Normalize Delay to the bus frequency): CYC = Delay / (Serial clock cycle in microseconds)
3. Round down CYC for an upper bound for the internal rate.
4. Repeat for up to 4 different bit rates per device, up to 16 bit rates in total.
5. Find an applicable common denominator for all rates of a device. Program the prescalar to divide with this number.
6. Program the sub rate counters.

Example 1:

Configure a 155.52-Mbps OC-3 PHY to 100 Mbps, for 8 bit UTOPIA bus at 25 Mhz (200 Mbps bus). 100 Mbps is equivalent to a cell every 106 cycles. In transmit internal rate mode, configure the sub rate divider to 105. This would pace the ATM scheduler at the prescribed rate.

Example 2:

Configure a 622-Mbps PHY and 32-Kbps PHY, for 16 bit UTOPIA bus at 50 MHz (800 Mbps bus).
622 Mbps is equivalent to a cell every 34.1 cycles.
Applying the formulation above, the process is:

1. Delay = 424/ 32E-3 = 13250
2. CYC = Delay/ 20E-3 = 622500
3. The only applicable common denominator for 34 and 622500 is 34.
4. Program the prescalar to divide by 34
5. Program sub rate 1 to divide by 1
6. Program sub rate 2 to divide by 18308

31.3.2.2 External Rate-Like Mode in PL2

The equivalent of an idle cell is a null packet. A null packet decrements the expiration counter from 1 to 0 for this PHY, and will be masked for a time set by its rate settings.

31.3.3 SPHY System Design

This section applies only to master mode.

31.3.3.1 Single-Device System

When the system includes only one SPHY device, clear the MultiPHY mode for the UCCs routed to the SPHY in the UPUC register (UPUCx[TMP] = 0). For a receive, clear MultiPHY mode in UPDCy[RMP]. The Clav must be a direct status (always valid).

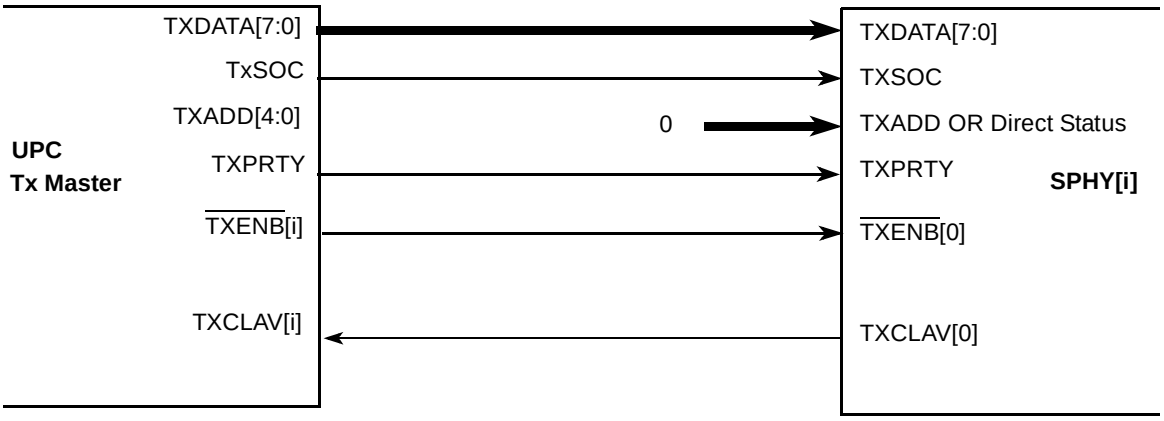


Figure 31-5. UPC with Tx UTOPIA SPHY System

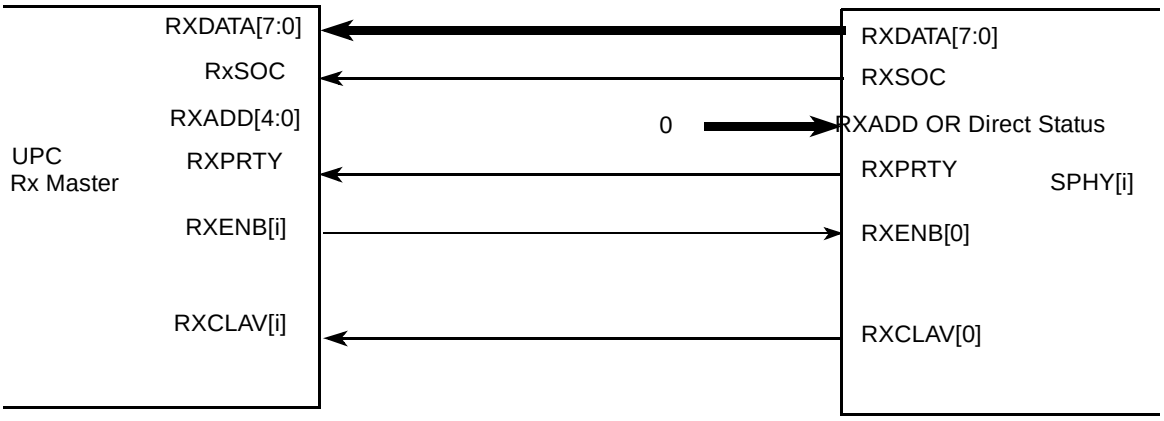


Figure 31-6. UPC with Rx UTOPIA SPHY System

31.3.4 SPHY Modes of Operation

The user can choose between a single-port or multi-port setting for a SPHY device. This mode affects how the user configures the transmit shaping for this device.

Table 31-1. SPHY Related Modes

Field Setting	Description	Meaning
UPDC[TMP]	0: Transmit Single PHY	A dedicated FIFO is assigned for the device routed to UCC
UPDC[TSP]	0: Single Port PHY 1: Multi Ports PHY	Single Port: QUICC Engine 1.0 block de-queues FIFO as a leaky bucket at internal rate. Single Scheduler for all channels. Multi Ports: QUICC Engine 1.0 block en-queues FIFO from multiple leaky buckets of different rates. Each port has its own scheduler. The FIFO is de-queued at the PHY request rate, so this mode can be considered as PHY's FIFO Full mode.
UPDC[RMP]	0: Receive Single PHY	The PHY is polled using an always-valid Clav/PRPA or a direct status Clav/ DRPA
UPDC[TB2B]	0: Switch to the next UCC in cyclic order. 1: Attempt to transmit in B2B	TB2B should be set for a SPHY with a rate higher then 50% of the bus bandwidth. If TSP=0, A B2B is conditional of a positive credit; if TSP=1, a B2B transmit is attempted as long as the PHY request data.
UPDC[RB2B]	0: Switch to the next Device in cyclic order. 1: Attempt B2B receive	RB2B should be set for a SPHY with a high bit rate in order to prevent PHY overrun.
UPDC[TPM]	1: Round robin (Fair) selection among pending qualified ports	Recommended if UPUC[TSP]=1

31.3.4.1 Single Port Mode

In Single-Port mode, the PHY is configured through port 0, and it must be enabled in the port enable register. The PHY has a single internal rate which should be set to the required transmit rate, and this controls the actual transmit rate. The UPC attempts to transmit a cell/packet on each expiration of the internal rate counter, and if a time lag has accumulated, it attempts to transmit cells at a faster rate until the

time lag has been compensated for. In ATM protocol, as a single PHY there is a single APC (multi-priority) transmit scheduling table for all the channels of this PHY. This mode is recommended when the line rate exceeds the required transmit rate, as it provides the least jitter on intercell/packet arrival time.

31.3.4.2 Multi-Port Mode

In Multi-Port mode, the PHY is represented by any number of ports (0–31), by enabling them in the respective port-enable register. Each port is a logical transmit end-point, with an internal rate and scheduler. Some users may want to direct traffic of different QoS and origin to different ports. This mode is recommended when the user wishes to bundle channels with similar QoS together, especially if the channels in each bundles have the same order bit-rate, but channels of different bundles have very high bit rate ratios.

31.3.5 Loopback Mode

The UPC supports loopback mode. The Rx and Tx UTOPIA signals are cross-connected internally. Output pins are driven; input pins are ignored. In loopback mode, transmitter and receiver operation must be complementary. That is, if the transmitter is master, the receiver is a slave and *vice versa*.

UPGCR must be configured for each Tx or Rx slave. Device 1 is used for both Rx and Tx operation and can be routed to any one of the three UCCs. Note that only one slave address can be configured in a MPHY system. For loopback connectivity of Tx and Rx, see Figure 31-7, Figure 31-8. If the slave is configured to MPHY mode, the master must be configured for MPHY mode. If the slave is configured to SPHY mode, there is no similar restriction on the master side, and the master side can be configured to MPHY or SPHY mode.

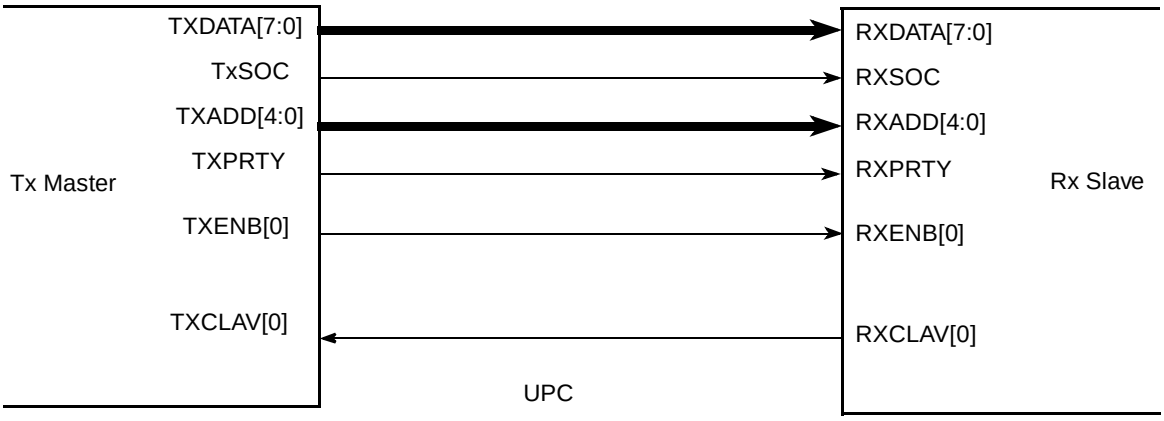


Figure 31-7. UTOPIA Tx Master to Rx Slave Internal Loopback System

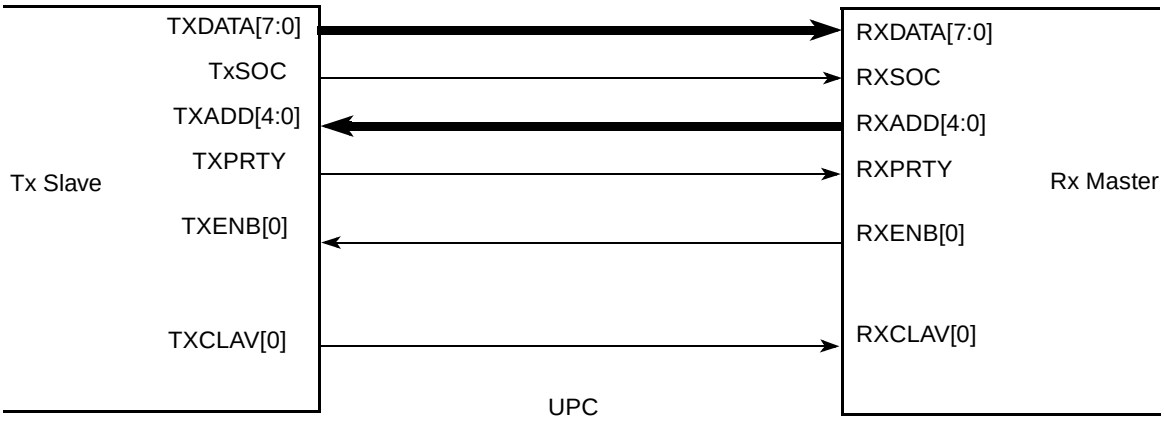


Figure 31-8. UTOPIA Tx Slave to Rx Master Internal Loopback System

31.3.5.1 Loopback Mode Programming Model Example

The example discussed in this section consists of a system with a Tx master connected to device 1 and an Rx slave on device 1. Tx master device 1 is routed to UCC1. Rx slave device 1 is routed to UCC3. This system can check several features of the UPC. If the Rx slave device is configured as an SPHY, it can respond to all addresses and simulate a full MPHY system. The programing model for the described system is shown in Table 31-2.

Table 31-2. Configurations for Loopback Example

Configuration	Bit Value	Description
UPC configurations	UPGCR[DIAG] = 01	Loopback mode
	UPGCR[ADDR] = 0	Default address multiplexing
	UPGCR[TMS] = 0	Tx master mode
	UPGCR[RMS] = 1	Rx slave mode
	UPLPA[Tx LAST PHY/PHY ID] = 31	Last PHY for MPHY Tx devices (for example)
UCC configurations	UPUC1[TMP] = 1	MPHY mode
Device configurations	UPDC1[Device Tx Routing] = 00	Route Tx device 1 to UCC1
	UPDC1[Device Rx Routing] = 01	Route Rx device 1 to UCC3

31.4 External Signals Description

NOTE: Slave Mode Signal Naming

Users familiar with the UTOPIA interface of the CPM in the MPC82xx and MPC85xx should know that the external signal naming convention of the QUICC Engine 1.0 block follows the UTOPIA standard. Therefore, naming in master and slave modes is different. The CPM retains the master mode signal naming for slave mode. For example, the QUICC Engine 1.0 block transmit SOC in slave mode is named RSOC but the CPM transmit SOC in slave mode is named TSOC.

In the QUICC Engine 1.0 block, users should connect signals between master and slave by name. In the preceding example, the user should connect the Master's TSOC with the QUICC Engine 1.0 block's TSOC.

This section describes the UPC external signals. One UTOPIA device configuration requires a total of 36 I/O ports for 8-bit mode. Table 31-3 summarizes all possible I/O pin assignments combinations.

Table 31-3. UCC UTOPIA I/O Pin Count

	UTOPIA 31 PHYs
8 bit Data	16
parity	2
Address	10
Clav/xPTA	2
Clock	2
Framing	2
Enable	2
Total for 8 bit	36

31.4.0.1 UTOPIA Interface Master Mode

UTOPIA master signals are shown in Figure 31-9.

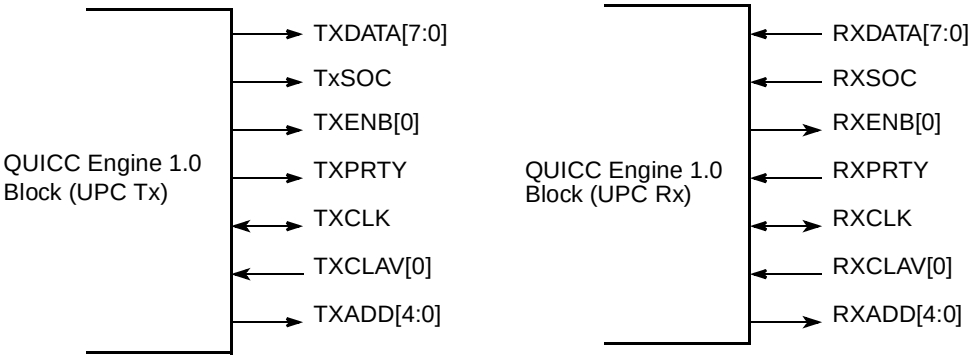


Figure 31-9. UTOPIA Master Mode Signals

Table 31-4 lists the I/O pins in UTOPIA Master mode.

Table 31-4. UTOPIA Master mode Signal Properties

Name	Function	I/O	Pull Up
TxDATA[7-0]	Transmit data from link layer to PHY	O	—
TxSOC	Transmit start of cell	O	—
$\overline{\text{TxENB}}$[0]	Transmit enable	O	—
TxCLAV[0]	Transmit cell available	I	—
TxPRTY	Transmit parity	O	—
TxCLK	Transmit clock	I	—
TxCLKO	Transmit clock out. Output to PHY.	O	—
TxADD[4-0]	Transmit address, TxADD[5] is the extra msb (TxAddr[4] for device B)	O	—
RxDATA[7-0]	Receive data from PHY to link layer	I	—
RxSOC	Receive start of cell	I	—
$\overline{\text{RxENB}}$[0]	Receive enable	O	—
RxCLAV[0]	Receive cell available	I	—
RxPRTY	Receive parity	I	—
RxCLK	Receive clock	I/O	—
RxADD[4-0]	Receive address, RxADD[5] is the extra msb (RxAddr[4] for device B)	O	—

31.4.0.2 UTOPIA Interface Slave Mode

UTOPIA slave signals are shown in Figure 31-10.

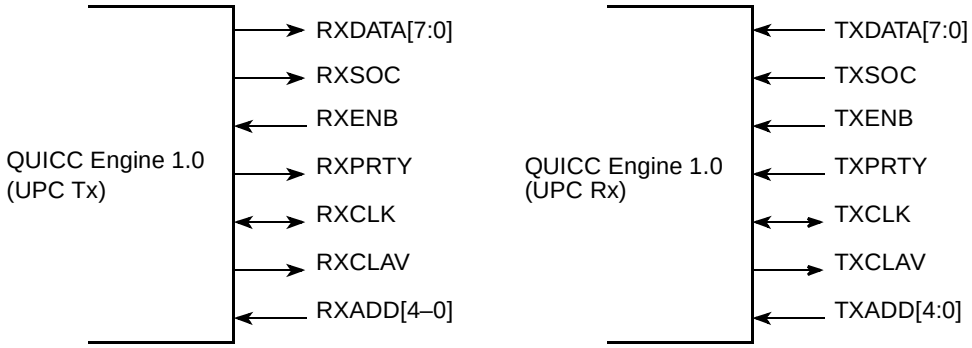


Figure 31-10. UTOPIA Slave Mode Signals

Table 31-5 lists the I/O pins in UTOPIA slave mode.

Table 31-5. UTOPIA Slave Mode Signal Properties

Name	Function	I/O	Pull Up
TxDATA[7–0]	Transmit data from PHY to link layer	I	—
TxSOC	Transmit start of cell	I	—
$\overline{\text{TxENB}}$[0]	Transmit enable	I	—
TxCLAV[0]	Transmit cell available	O	—
TxPRTY	Transmit parity	I	—
TxCLK	Transmit clock	I	—
TxCLKO	Transmit clock	O	—
TxADD[4–0]	Transmit address	I	—
RxDATA[7–0]	Receive data from link layer to PHY	O	—
RxSOC	Receive start of cell	O	—
$\overline{\text{RxENB}}$[0]	Receive enable	I	—
RxCLAV[0]	Receive cell available	O	—
RxPRTY	Receive parity	O	—
RxCLK	Receive clock	I/O	—
RxADD[4–0]	Receive address	I	—

31.5 Memory Map

This section provides a memory map and detailed descriptions of registers.

Table 31-6. UPC Register Summary

Offset[1]	Register	Access	Reset Value	Size (bytes)
General configuration Registers				
0x0	UPGCR - General Config	R/W	0x0000_0000	1
0x4	UPLPA - Last PHY Addr	R/W	0x0000_0000	2
0x8	UPHEC - UL2 HEC config/ PL2 Rx Segment Size	R/W	0x0000_0000	2
0xC	UPUC - UCC Configuration	R/W	0x0000_0000	4
0x10	UPDC1 - Device1 Config	R/W	0x0000_0000	4
0x14	Reserved	R/W	0x0000_0000	4
0x18	Reserved	R/W	0x0000_0000	4
0x1C	Reserved	R/W	0x0000_0000	4
0x20	Reserved	R/W	0x0000_0000	1

Table 31-6. UPC Register Summary (continued)

Offset[1]	Register	Access	Reset Value	Size (bytes)
0x21–0x2F	Reserved			15
Port Registers				
0x30	UPDRS1_H - Device1 Rate Select	R/W	0x0000_0000	4
0x34	UPDRS1_L - Device1 Rate Select	R/W	0x0000_0000	4
0x38	Reserved	R/W	0x0000_0000	4
0x3C	Reserved	R/W	0x0000_0000	4
0x40	Reserved	R/W	0x0000_0000	4
0x44	Reserved	R/W	0x0000_0000	4
0x48	Reserved	R/W	0x0000_0000	4
0x4C	Reserved	R/W	0x0000_0000	4
0x50	UPDRP1 - Device1 Receive Priority	R/W	0x0000_0000	4
0x54	Reserved	R/W	0x0000_0000	4
0x58	Reserved	R/W	0x0000_0000	4
0x5C	Reserved	R/W	0x0000_0000	4
Event Registers				
0x60	UPDE1 - Device1 Event	R/W	0x0000_0000	4
0x64	Reserved	R/W	0x0000_0000	4
0x68	Reserved	R/W	0x0000_0000	4
0x6C	Reserved	R/W	0x0000_0000	4
Internal Rate Registers				
0x70	UPRP1 - Device 1Internal Rate configuration	R/W	0x0007	2
0x72	Reserved	R/W	0x0007	2
0x74	Reserved	R/W	0x0007	2
0x76	Reserved	R/W	0x0007	2
0x80	UPTIRR1_0 - Device1 Transmit Internal Rate 0	R/W	0x0000_0000	2
0x82	UPTIRR1_1 - Device1 Transmit Internal Rate 1	R/W	0x0000_0000	2
0x84	UPTIRR1_2 - Device1 Transmit Internal Rate 2	R/W	0x0000_0000	2
0x86	UPTIRR1_3 - Device1 Transmit Internal Rate 3	R/W	0x0000_0000	2
0x88	Reserved	R/W	0x0000_0000	2
0x8A	Reserved	R/W	0x0000_0000	2
0x8C	Reserved	R/W	0x0000_0000	2
0x8E	Reserved	R/W	0x0000_0000	2
0x90	Reserved	R/W	0x0000_0000	2

Table 31-6. UPC Register Summary (continued)

Offset[1]	Register	Access	Reset Value	Size (bytes)
0x92	Reserved	R/W	0x0000_0000	2
0x94	Reserved	R/W	0x0000_0000	2
0x96	Reserved	R/W	0x0000_0000	2
0x98	Reserved	R/W	0x0000_0000	2
0x9A	Reserved	R/W	0x0000_0000	2
0x9C	Reserved	R/W	0x0000_0000	2
0x9E	Reserved	R/W	0x0000_0000	2
0xA0	UPER1 - Device 1 Port Enable	R/W	0x0000_0000	4
0xA4	Reserved	R/W	0x0000_0000	4
0xA8	Reserved	R/W	0x0000_0000	4
0xAC	Reserved	R/W	0x0000_0000	4
0xB0-0xFF	Reserved	—	—	—

Note:

1 0x2E00 for UPC1, 0x3E00 for UPC2

31.6 UPC Register Descriptions

31.6.1 UPC General Configuration Register (UPGCR)

Table 31-7 describes the UPGCR fields

Table 31-7. UPGCR Register Field Descriptions

Bits	Name	Description
0	Protocol	0 UL2 1 Not applicable
1	TMS	Transmit master/slave mode 0 Transmit master mode is selected. 1 Transmit slave mode is selected.
2	RMS	Receive master/slave mode 0 Receive master mode is selected. 1 Receive slave mode is selected.
3	ADDR	Master MPHY Addr multiplexing: (Ignored for slave) 0 5-bit addressing MPHY device operation (default). 31 addresses with separate Clav/Enb for each device. Parallel polling in multi device system. 1 Not applicable
4–6	Reserved	Must be cleared.

Table 31-7. UPGCR Register Field Descriptions (continued)

Bits	Name	Description
7	DIAG[1]	Diagnostic mode 0 Normal mode 1 Loopback mode In Loopback, UTOPIA Rx and Tx signals are shorted internally. Output pins are driven, input pins are ignored.
8–31	—	Not in use

Note:

1 Note on backwards compatibility: the programing model regarding the LPB mode have changed, instead of programing the GUMR[DIAG] as in the 8260, it is done in through UPGCR[DIAG].

31.6.2 UPC Last PHY Address Register (UPLPA)

Table 31-8 describes the UPLPA fields.

Table 31-8. UPLPA Register Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3–7	Tx LAST PHY/PHY ID	Master mode: Tx Last PHY. This is the last polled PHY addr (MPHY Master mode only) 5 lsb. The 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). Default MPHY mode: Maximal value = 31 6-bit addressing MPHY mode: Maximal value = 15 (Addr[4] is not valid). Common value for device pairs A,B. Slave mode: Tx PHY address
8–10	—	Reserved, should be cleared.
11–15	Rx LAST PHY/PHY ID	Master mode: Rx Last PHY. The last polled PHY addr (MPHY master mode only) 5 lsb. The 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. Default MPHY mode: Maximum value = 31 6-bit addressing MPHY mode: Maximum value = 15 (Addr[4] is not valid). Common value for device pairs A, B. Slave mode: Rx PHY address
16–31	—	Not in use

31.6.3 UPC HEC Register (UPHEC)

Table 31-9 describes the UPHEC fields.

Table 31-9. UPHEC Register Field Descriptions

Bits	Name	Description
0–7	HEC_HI	UTOPIA: For 8 bit UTOPIA, the value of this field is sent instead of the HEC on TxData[7:0]. Note: HEC_HI is a constant byte for the HEC. It does not generate the HEC; instead it only outputs this constant byte as a 'placeholder' for the HEC. This byte is then replaced by the ATM PHY with the actual value
8–15	HEC_LO	For 8 bit UTOPIA this field is ignored.
16–31	—	Not in use

31.6.4 UPC UCC Configuration Register (UPUC)

Table 31-10 describes the UPUCx fields.

Table 31-10. UPUC Register Field Descriptions

Bits	Name [1]	Description
0, 8, 16, 24	TMP_x	Transmit Multiple PHY mode. This mode should be set when multiple devices or a device with several PHYs is routed to this UCC. UPC Tx Master mode: 0 SPHY mode: Transmit to SPHY device (Tx FIFO is attempted to be filled. Clav/PTA is always assumed valid[2], PHY address must be 0. No other device can be routed to the UCC specified in the routing field. 1 MPHY mode: Transmit to MPHY device/s UPC Tx Slave mode: 0 SPHY system (Output Clav is a direct status, do not release outputs to high impedance) 1 MPHY system (Output Clav is a response for polling, release outputs to high impedance)
1, 9, 17, 25	TSP_x	Transmit Single PHY mode. This mode is valid only if TMP=0: 0 Single EP: Internal address of this PHY must be 0. As Master, the Internal rate pace the de-queuing of the FIFO. As slave, the internal rate enable the assertion of Clav when the FIFO is not empty. 1 Multi EP: Internally behaves as a multi-port PHY (multiple End Points): the Internal rate pace the request.rate for each EP. As master, FIFO is de-queued based on Clav status from the PHY. As slave, Clav reflect the FIFO condition (negated when FIFO is empty).
2, 10, 18, 26	$TB2B_x$	Transmit back to back (valid only in master mode). 0 Do not attempt Transmit B2B from this UCC. 1 Attempt Transmit B2B from this UCC. Note: Should be cleared.
3–7, 11–15, 19–23, 27–31	Reserved	Reserved, should be cleared.

Note:

1 x is the UCC index. For UPC1, x=0-3 for UCC1,3,5,7. For UPC2, x=0-3 for UCC2,4,6,8 respectively.

2 Either direct status or the address lines of the PHY are hard wired fixed to the PHY's address.

31.6.5 UPC Device X Configuration Register (UPDCx)

31.6.5.1 UPDCx in ATM Protocol

Table 31-11 describes the UPDCx fields.

Table 31-11. UPDCx in ATM Protocol Field Descriptions

Bits	Name	Description
0–3	TEHS	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[1]	Idle cells discard 0 Discard idle cells (GFC, VPI, VCI, PTI =0). 1 Do not discard idle cells.
9–10	PE	Port enable 00 Master: UPER has no affect: All ports are enabled. Slave: UPER has no affect, port0 is enabled regardless of Phy address. 01 UPER masks only Rx 10 UPER masks only Tx 11 UPER masks both Rx and Tx
11	RES	Must be cleared
12–13	TxUCC	00 UCC1 (UPC1) or UCC2 (UPC2) 01 UCC3 (UPC1) or UCC4 (UPC2) 10 UCC5 (UPC1) 11 Reserved
14–15	RxUCC	00 UCC1 (UPC1) or UCC2 (UPC2) 01 UCC3 (UPC1) or UCC4 (UPC2) 10 UCC5 (UPC1) 11 Reserved
16	Tx Enb	0 Tx Disabled 1 Tx Enabled
17	Rx Enb	0 Rx Disabled 1 Rx Enabled
18	RB2B	Receive back to back (valid only in master mode). 0 Do not attempt Receive B2B from this device. 1 Attempt B2B receive from this device. Can be used in order to enable a PHY back to back on a SPHY, or prioritize a device. Note: Should be cleared.
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.

Table 31-11. UPDCx in ATM Protocol Field Descriptions (continued)

Bits	Name	Description
21	RXP	Receive parity check. 0 Check Rx parity line. 1 Do not check Rx parity line.
22–23	—	Reserved
24	TXD port width	Transmit data bus width Device/SPHY 0 8-bit data bus width 1 16-bit data bus width
25	RXD port width	Receive data bus width Device/SPHY 0 8-bit data bus width 1 16-bit data bus width
26	TPM	Tx Selection Priority Mode. This controls the priority by which PHYs are serviced. 0 Round Robin. After a PHY has been selected, the selection resumes from the next PHY. 1 Fixed. For each selection PHY0 has the highest priority and PHY31 the lowest priority. (The internal rate prevents starvation of high ID PHYs).
27	Reserved	Must be cleared
28	RMP	Receive Multiple PHY mode UPC Rx Master mode: 0 Receive from SPHY device (Clav/PTA is always assumed valid[2], PHY address must be 0) 1 Receive from MPHY device/s UPC Rx Slave mode: 0 SPHY system 1 MPHY system
29	HECI[3]	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[4]	HEC Check 0 Do not check Rx HEC 1 Check Rx HEC
31	COS[5]	Coset mode on the HEC

Note:

1 Discard Rx Idle cell.

2 Either direct status or the address lines of the PHY are hardwired to the PHY address.

3 UDC HEC include (Rx and Tx).

4 HEC Check (HW HEC check).

5 Coset mode on the HEC (HW HEC XOR with 0x55).

31.6.6 UPC Device X Internal Rate Configuration Register (UPRP*x*)

Offset: UPRP1: 0x70,
UPRP2: 0x72,
UPRP3: 0x74,
UPRP4: 0x76

Access: User Read Only

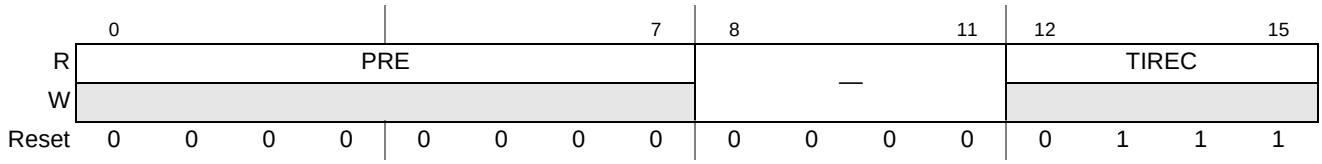


Figure 31-11. Device X Internal Rate Configuration Register (UPRP*x*)

Table 31-12 describes the UPRP*x* fields.

Table 31-12. UPRP*x* Field Descriptions

Bits	Name	Description
0–7	PRE	Sets the prescalar divider value. Prescalar divider value is PRE+1 00x: Divide by 1 0xFF: Divide by 256
8–11	Reserved	Must be cleared
12–15	TIREC	Transmit Internal/External Rate Expiration Counter for device X. Values: 0 External Rate: Transmit rate is limited by Clav, polling and selection priorities. (Idle cells are discarded and not transmitted). 1 Internal Rate, PPS mode. No burst and no underrun error indication. Recommended for PON. 2–15 Internal rate with burst and underrun indication (For ATM protocols only). 7 Default value (82xx internal rate compatible) This field sets the maximum allowed cell credit for the PHYs. An internal rate expiration counter holds the credit balance per PHY. It increments every time the internal rate expires. It decrements every time a cell is transmitted to the PHY (including idle cells, which are not actively transmitted). While the credit is positive (non zero), the UPC transmits cells to that PHY. When the credit reaches the value of the TIREC, a transmit underrun event is set in registers UPDEx.

31.6.7 UPC Device X Transmit Internal Rate Register (UPTIRR*x*_*y*)

Offset: UPTIRR1_1 = 0x80, UPTIRR1_2 = 0x82, UPTIRR1_3 = 0x84, UPTIRR1_4 = 0x86
UPTIRR2_1 = 0x88, UPTIRR2_2 = 0x8A, UPTIRR2_3 = 0x8C, UPTIRR2_4 = 0x8E
UPTIRR3_1 = 0x90, UPTIRR3_2 = 0x92, UPTIRR3_3 = 0x94, UPTIRR3_4 = 0x96
UPTIRR4_1 = 0x98, UPTIRR4_2 = 0x9A, UPTIRR4_3 = 0x9C, UPTIRR4_4 = 0x9E

Access: User Read Only

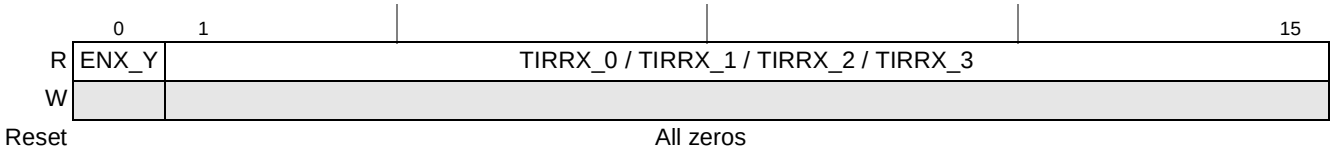


Figure 31-12. UPC Device X Transmit Internal Registers (UPTIRR*x*_*y*)

Table 31-13 describes the UPTIRRx fields.

Table 31-13. UPTIRR Register Field Descriptions

Bits	Name	Description
0	ENX_Y	0 Device X, Sub Rate Divider is disabled 1 Device X, Sub Rate Divider is enabled
1–15	TIRRX_Y	Device X, Sub rate divider Y is prescaled clock divided by <TIRRX_Y>+1

31.6.8 UPC Device X Port Enable Register (UPERx)

Table 31-14 describes the UPERx fields.

Table 31-14. UPC Device X Port Enable Register (UPER)

Bits	Name	Description
0	PE0	Port0 Enable: The port enable option depends on UPDC[PE] field. 0 Port0 is disabled. Clav/PxPA from this port is internally masked. 1 Port is enabled. For slave: 0 Slave is disabled 1 Slave is enabled (refer also to PE<n>)
1–31	PE<n>	Port <n> Enable. The port enable option depends on UPDC[PE] field. 0 Port is disabled. Clav/PxPA from this port is internally masked. 1 Port is enabled. For slave in MPHY system: 0 Slave will not respond to polling of address <n> 1 Slave will respond to polling of address <n> For slave in MEP mode: 0 Endpoint is disabled. 1 Endpoint is enabled.

31.6.9 UPC Device X Transmit Rate Select Register (UPDRS*x*)

Table 31-15 describes the UPDRSx fields.

Table 31-15. UPDRS Register Field Descriptions

Bits	Name	Description
<2*n:2*n+1>	RSP<n>	Port <n> Transmit Rate Select[1] 00 TIRRx0 01 TIRRx1 10 TIRRx2 11 TIRRx3

Note:

1 In 6-bit addressing MPHY mode, ports 0–15 correspond to ports of device A; ports 16–31 correspond to ports 0–15 of device B.

31.6.10 UPC Device X Receive Port Priority Register (UPDRP*x*)

Table 31-16 describes UPDRPx fields.

Table 31-16. UPDRP Register Field Descriptions

Bits	Name	Description
<n>	RPP<n>	Port <n> Receive Priority[1] 0 RPx0 (High Priority) 1 RPx1 (Low Priority)

Note:

1 In 6-bit addressing MPHY mode, ports 0 –15 correspond to ports of device A; ports 16–31 correspond to ports 0–15 of device B.

31.6.11 UPC Device X Event Register (UPDE*x*)

Table 31-17 describes the UPDEx fields.

Table 31-17. UPDE Register Field Descriptions

Bits	Name	Description
<n>	EV<n>	Port <n> Event[1], as configured by UPGCR[Status] mode. The event is a sticky bit set by the UPC; the host clears it by writing a 1 to it. Writing 0 has no effect. Transmit IR Underrun. The transmit internal rate expiration counter has reached the threshold set by TIREC. TIRU can be set only when a TIREC value larger than 1.0 is used. 0 No underrun 1 Underrun occurred TVCPA: Transmit Valid Cell/Packet Available: 0 A valid TxClav or PTPA from Port <n> was not detected since the last time this bit was cleared. 1 A valid TxClav or PTPA from Port <n> was detected since the last time this bit was cleared. RVCPA: Receive Valid Cell/Packet Available: 0 A valid RxClav or PRPA from Port <n> was not detected since the last time this bit was cleared. 1 A valid RxClav or PRPA from Port <n> was detected since the last time this bit was cleared.

Note:

1 In 6-bit addressing MPHY mode, ports 0 to 15 correspond to Ports of device A, Ports 16 to 31 correspond to Ports 0 to 15 of device B.

31.6.12 UCCx Event/Mask Registers (UCCEx, UCCMx)

Access: User Read Only

	0	4	5	6	7	8	9	10	11	12	13	14	15
R	—		TIRU	GRLI	GBPB	GINT3	GINT2	GINT1	GINT0	INTO3	INTO2	INTO1	INTO0
W													
Reset	All zeros												

Figure 31-13. UCCE*x*/UCCM*x* in ATM Protocol

31.7 Functional Description: Receive Priority Among Pending PHYs

This section describes how priority selection is handled in the receiver. This feature is required in addition to policing to allocate enough bandwidth to fast PHYs over slow PHYs.

Polling address generation always proceeds in a periodic order.

The user can assign a weight to different PHYs in the Rx scheduling to prevent overrun of fast PHYs by multiple slow PHYs.

Each PHY is assigned one of two priority levels. A round robin selection scheme is used among all the pending PHYs of the same priority. The priority is selected by register UPDRPx. Configure UPDRPx[RPP] = 0 for fast PHYs and UPDRPx[RPP] = 1 for slow PHYs.

31.8 Glossary

Device	A group of PHYs or ports (could be also a single PHY), controlled by common control signal pair (xCLAV/PxPA and xEnb).
Expiration Counter	Holds the state of the jitter buffer per PHY (counts the credit for a PHY). This counter controls the transmit shaping. It must be positive to fetch a cell/packet to the transmit queue. A PHY's CLAV/xPTA is qualified only when the expiration counter of the PHY is positive. It is incremented on each expiration of the matching Internal Rate Timer while there is a valid request from the PHY, and it is decremented each time a cell/packet is dequeued. An overflow of the counter causes a Transmit Internal Rate Underrun event to be generated.
Internal Rate	The scheduling of transmit request per PHY. Shapes traffic by allocating a maximal bus bandwidth per PHY. This is the lowest level of traffic shaping.
Internal Rate Underrun	The event of an underrun of a conceptual transmit jitter buffer. This is the event of a mismatch of up to 16 cells/packets[1] over any interval of time between the number of PHY transmit requests (which are polled at the Internal Rate) and the actual number of transmitted cells/packets (per PHY). See also Expiration Counter.
Multi Device	Multiple devices on the bus controlled by separate Clav/Enb signals, and optionally by a separate xADDR[MSB] (see Extended address). Not applicable to 832x.

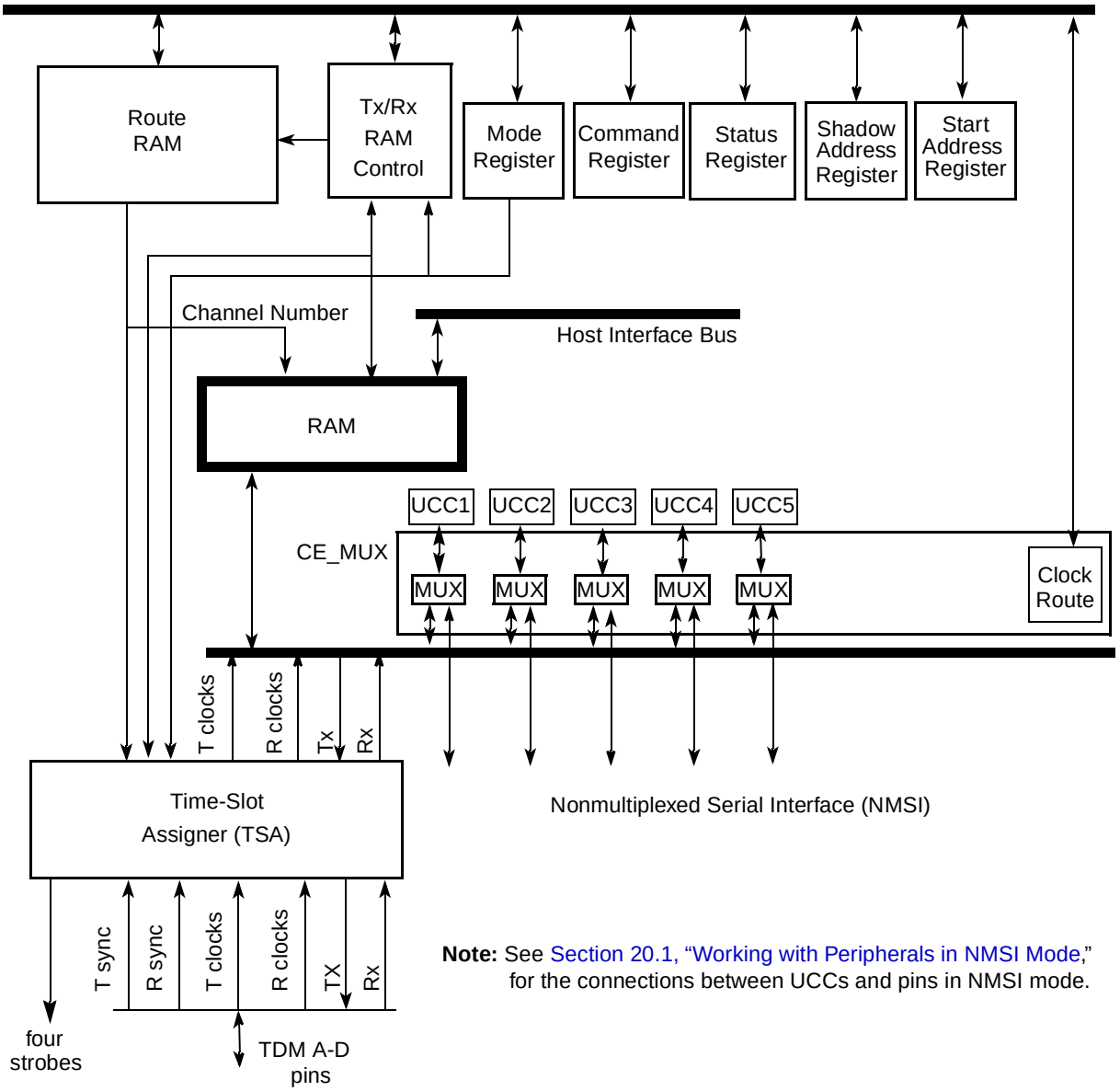
1. Could be less then 16, the size of the jitter window is configurable.

PHY	A physical bus endpoint, usually has a line connection. A PHY could be represented by several ports, for example in ADSL fast and interleaved paths. The UPC does not distinguish between a PHY and a Port.
Polling	The mechanism which allows the master to poll the PHYs to know if there is a cell/packet available for Rx or if the PHY can accept another cell/packet for Tx.
Port	A logical bus endpoint, which has a unique PHY address (see also PHY)
Rx	Receive operation, master transfer cell/packet from the PHY.
Routing	Usually the assignment of devices to UCCs. UPC1 Can be routed to UCC1, UCC3, UCC5. UPC2 can be routed to UCC2, UCC4.
Selection	The mechanism which selects a specific PHY (from those that were successfully polled) for Tx or Rx.
Single PHY (SPHY)	A device with a single PHY (and a single port). In this mode it is assigned a dedicated FIFO (recommended for fast PHYs—for example, an uplink PHY).
Sub rate counter	The mechanism to pace the Internal Rate. Each device has 4 Transmit Internal rate timers (The UPC support a total of 16 different internal rates). Each PHY can use a different timer.
Tx	Transmit operation, master transfer cell/packet to the PHY.
UCC	Unified Communication Controller, which defines the protocol and control the data and status FIFOs of the UPC. The UPC can be connected to up to 4 UCCs.

Chapter 32
Serial Interface with Time-Slot Assigner

32.1 Serial Interface Block Diagram

The serial interface (SI) block diagram is shown in Figure 32-1.



Note: See Section 20.1, "Working with Peripherals in NMSI Mode," for the connections between UCCs and pins in NMSI mode.

Figure 32-1. SI Block Diagram

32.2 Overview

The serial interface (SI) manages the routing of four time-division multiplexing (TDM) lines to the QUICC Engine block serial drivers, the UCCs, for receive and transmit. TDM is technique used in data communications for combining several lower-speed channels into one transmission path at a higher speed in which each low-speed channel is assigned a specific position based upon time in the signal stream.

The SI has four time-slot assigner (TSA) modules, each handling one TDM line. Each TSA can route time slots to any UCC. The SI stores the routing entries in a RAM. 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. Various TDM configurations are shown in Figure 32-2 through Figure 32-5. TSA programming is independent of the protocol used by the UCC. For instance, the fact that UCC2 can be programmed for the HDLC protocol does not affect TSA programming.

The TSA implements both internal route selection and time-division multiplexing for multiplexed serial channels. The TSA supports the serial bus rate for most standard TDM buses, including T1 and CEPT highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates.

Each TDM can support E3 or DS-3 rates as a clear channel in either a parallel-nibble or serial interface.

NOTE: On Backward-Compatibility

GCI mode is not supported in the QUICC Engine block.

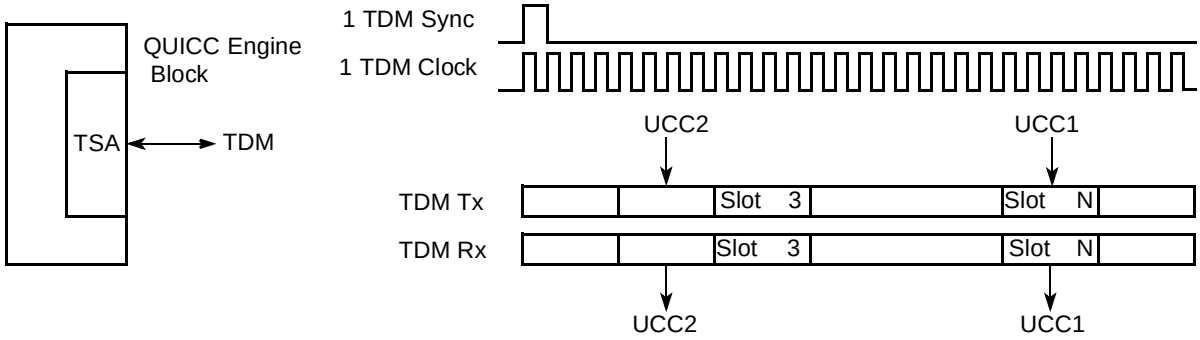


Figure 32-2. Simple TDM Example

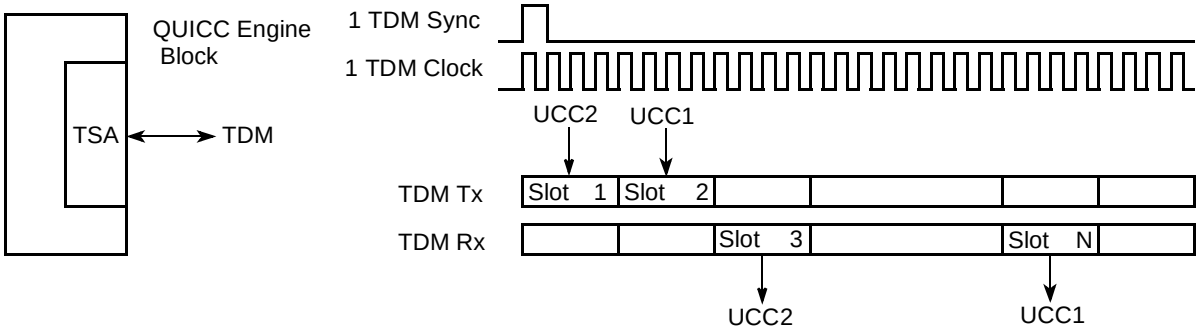
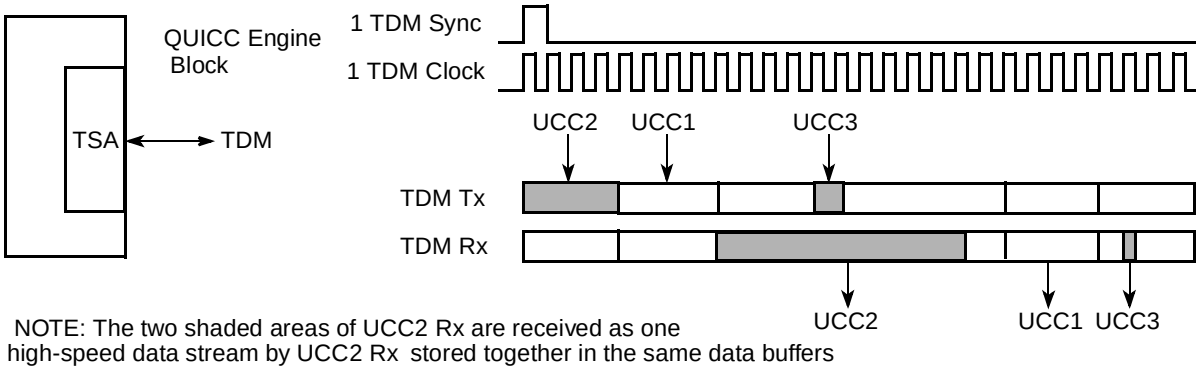


Figure 32-3. TDM Example—Different Tx and Rx Routing



NOTE: The two shaded areas of UCC2 Rx are received as one high-speed data stream by UCC2 Rx stored together in the same data buffers

Figure 32-4. TDM Example—Multiple Time Slot Per Channel With Varying Sizes of Time Slot

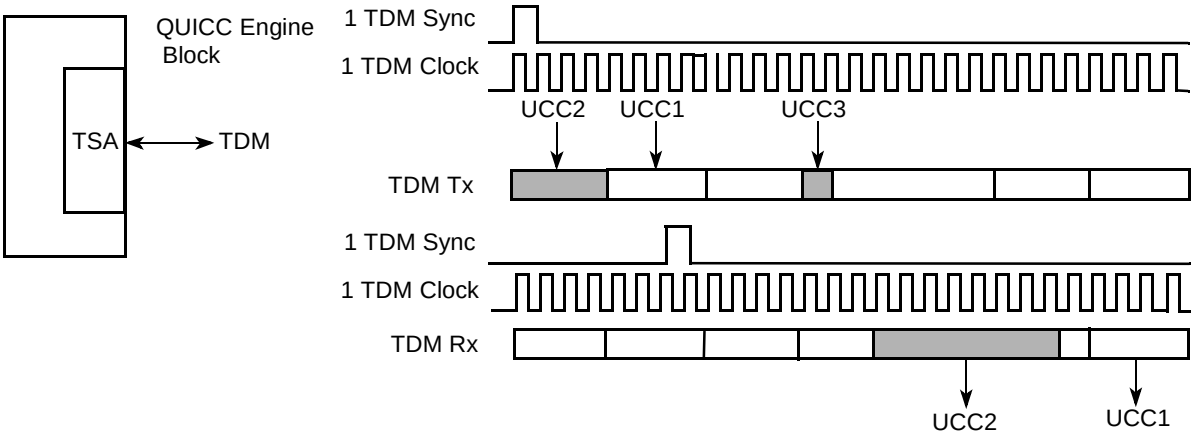
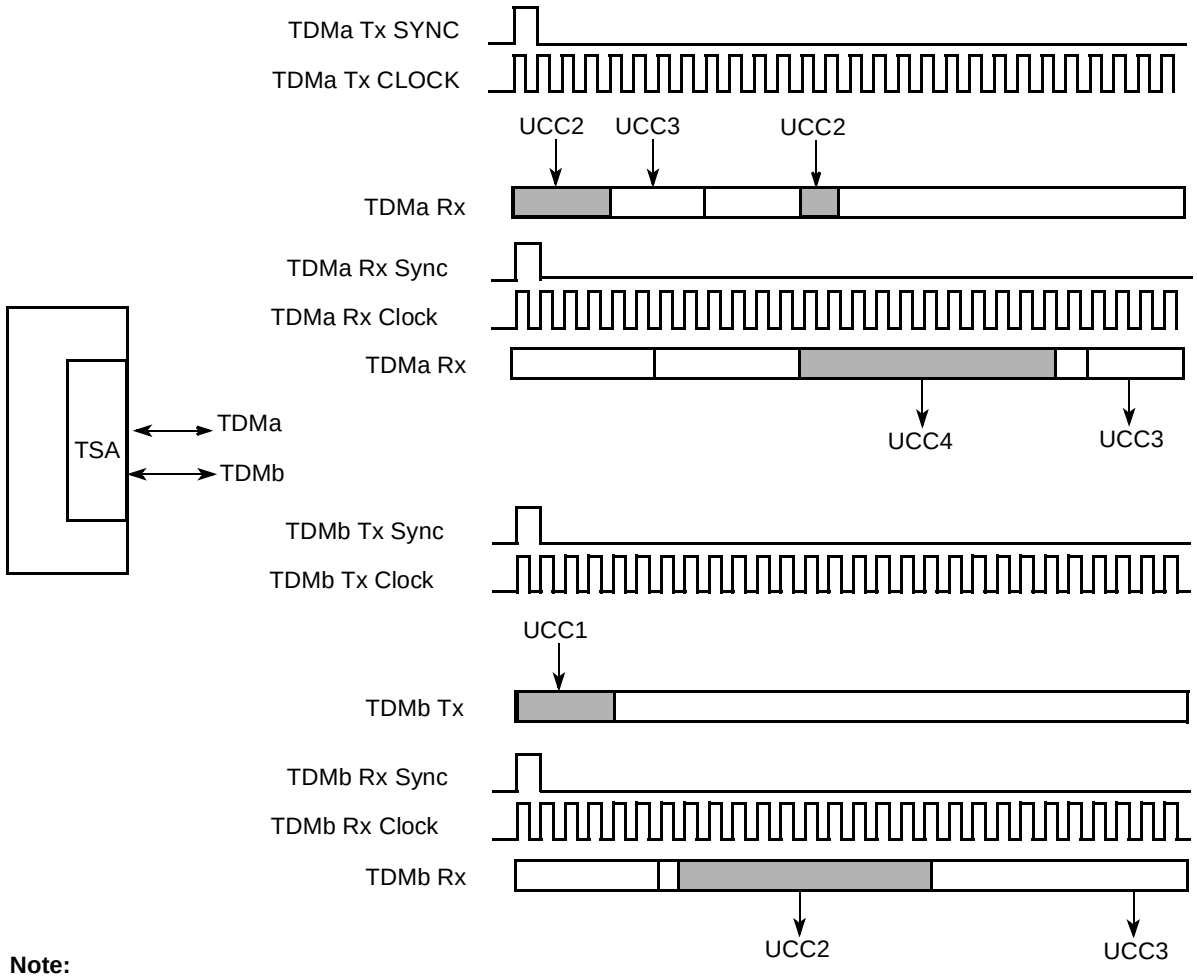


Figure 32-5. TDM Example—Totally Independent Rx and Tx

At its most flexible, the TSA can provide four separate TDM channels, each with independent receive and transmit routing assignments and independent sync pulse and clock inputs. Thus, the TSA can support eight, independent, half-duplex TDM sources, four in reception and four in transmission, using four sync inputs and four clocks each. A dual-channel example is shown in Figure 32-6.



Note:
UCCs can receive on one TDM and transmit on another (UCC2 and UCC3).

Figure 32-6. 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 (per routing entry). These strobes are completely independent from the channel routing used by the UCCs. 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. Note that several strobes may be driven for the same frame. Each strobes can be used by each channel, but care must be taken if using the same strobe by two different TDMs.

Most TSA programming is done in two of 512 entries × 16-bit SI RAMs (receive and transmit). These SI RAMs are directly accessible by the host processor in the chip. One SI RAM is always used to program the transmit routing; the other is always used to program the receive routing. SI RAMs can be used to define the number of bits/bytes to be routed to the UCC and determine when external strobes are to be asserted and negated.

The number of SI RAM entries available for time-slot programming depends on the user's configuration. For each TDM the user defines how many of the 512 entries are used by that TDM. If on-the-fly changes are allowed, the SI RAM entries are reduced according to the user's programming.

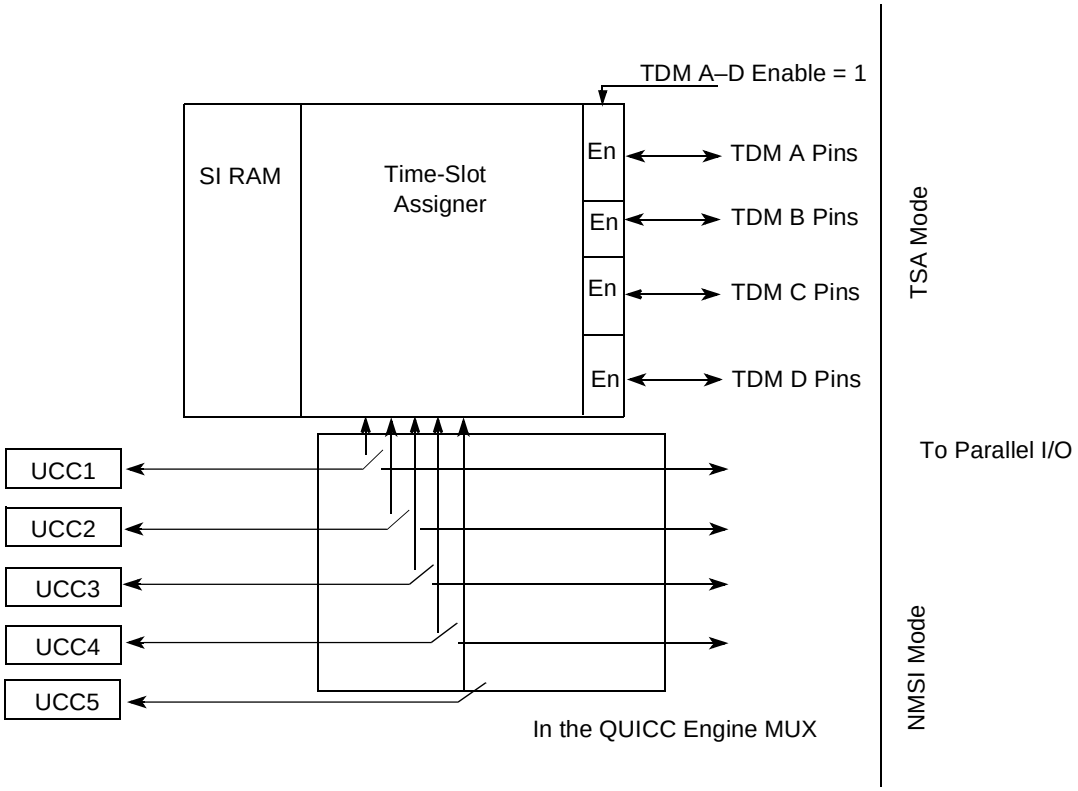
The SI supports two testing modes—echo and loopback.

- 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 UCC 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 the SI and the internal channel routes. When applying external/internal loopback with SIxMR[CRT]= 0, rx/tx synchronization is kept by one of the following conditions:
  - First RSYNC/TSYNC is asserted 10 serial clocks after TDM is enabled.
  - Only one RSYNC/TSYNC is asserted per frame.

Note that the flexibility described in the preceding section can be applied to each of the four TDM channels and to all UCC's interfaces independently.

32.2.1 Enabling Connections to TSA

Each UCC can be independently enabled to connect to TSA or dedicated external pins (NMSI).The four TDMs are connected to four independent TDM interfaces. The connection between the TSA and the serial interfaces is shown in Figure 32-7. The connection is made by programming the CE_MUX and the parallel I/O. After the connections are made, the exact routing decisions are made following the SI RAM entries.



Note: The connections between UCCs and pins in NMSI mode are not shown in this figure.

Figure 32-7. Enabling Connections to the TSA

32.3 Features

The SI includes the following features:

- Can connect to four independent TDM channels. Each can be one of the following:
  - T1 or CEPT line
  - Integrated services digital network primary rate (PRI)
  - ISDN basic rate–interchip digital link in 4 channels (IDL)
  - E3 or DS3 clear channel.
  - User-defined interfaces.
- Independent, programmable transmit and receive routing paths.
- Total of 512 routing entries for receive and transmit each.
- Total of 256 routing entries + 256 shadow routing entries for receive and transmit each.
- Common or independent transmit and receive frame syncs allowed.
- Common or independent transmit and receive clocks allowed.
- Selection of rising/falling clock edges for the frame sync and data bits.
- Selectable delay (0–3 bits) between frame sync and frame start.

- Four programmable strobe outputs and four clock output pins.
- 1- or 8-bit resolution in routing, masking, and strobe selection.
- Internal routing and strobe selection can be dynamically programmed.
- Loopback or echo per bit.
- Switch RxD and TxD lines per bit.
- Supports automatic echo and loopback mode for each TDM.
- Supports parallel-nibble interface for E3 or DS3 on each TDM.
- Can route any of 0–128 time slots to any of 5 UCCs.

NOTE: On Backward-Compatibility

- Backward-compatibility for the programming model is kept in the SI except for the following cases: GCI normal mode is not supported in the QUICC Engine block.

NOTE: Enhancements

- Base address for current or shadow RAM in 16 entries granularity.
- Enhanced debug mode: Switch transmit and receive, loopback/echo per entry for each UCC.

32.4 Modes of Operation

The SI has three main operating modes for the TDMs:

- Static routing. The TDMs routing definitions in RAM can be changed only when the TDM is disabled. After enabling the TDM the new definitions will take place. (The entire RAM may be used but not for shadow RAM).
- Dynamic routing. The TDMs routing definitions can be modified while the TDM is enabled, by programming a different routing RAM and using a host command to switch the routing RAM. (Part of the RAM is used for shadow configuration).
- Multi-frame: The frame structure looks like a two repeating loops. The first loop executes from the TDM's current SI RAM and the second loop from the shadow RAM. The number of iterations per loop is programmable. This mode is used to compress the RAM size required for long frames.

In addition each mode has two testing modes—echo and loopback.

- 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 serials echo mode in that it can operate on the entire TDM signal rather than just on a particular serial channel.
- Loopback mode cause 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 route.

32.5 SI External Signal Descriptions

This section defines the SI to TDM I/O pins. SI external signal list is shown in Table 32-1.

Table 32-1. SI External Signal Table

Name	Function	I/O
SI:STRB[1:4]	SI strobe signal	O
TDM[A:D]:CLKO	clock out for TDM[A:D]	O
TDM[A:D]:RSYNC	Receive data sync TDM[A:D]	I
TDM[A:D]:RXD[0:3]	Receive data TDM[A:D]	I
TDM[A:D]:TSYNC	Transmit data sync TDM[A:D] or IDL grant.	I
TDM[A:D]:TXD[0:3]	Transmit data TDM[A:D]	O
TDM[A:D]:RQ	IDL request permission to transmit on D channel	O

32.5.1 SI Detailed Signal Descriptions

Detailed information about SI external signal list is shown in Table 32-2.

Table 32-2. Interface A—Detailed Signal Descriptions

Signal	I/O	Description	
SI:STRB[1:4]	O	SI strobe signal. The strobe signals may be needed for interfacing other devices.	
		State Meaning	Asserted—Strobe signal is set. Negated—strobe signal is reset.
		Timing	Assertion—in 0-bit delay frame - May occur at any time, otherwise synchronous to clock edge according to SIMR FE & CE fields. Negation—in 0-bit delay frame - May occur at any time, otherwise synchronous to clock edge according to SIMR FE & CE fields.
TDM[A:D]:CLKO	O	clock out for TDM[A:D].This clock is in a rate of 1:1 regarding TDM's receive clock.	
		State Meaning	Asserted / Negated— TDM's receive clock: Data is sampled on clock's rising edge.
		Timing	Assertion / Negation—according to TDM's receive clock.
TDM[A:D]:RSYNC	I	Receive frame sync from TDM [A:D].	
		State Meaning	SIxMR[SL] is cleared: Rising edge of RSYNC will trigger start of frame according to delay in SIxMR[RFSD]. In-frame rising edge is ignored. SIxMR[SL] is set: Falling edge of RSYNC will trigger start of frame according to delay in SIxMR[RFSD]. In-frame falling edge is ignored.
		Timing	Assertion / Negation— Sampled or latched on TDM Rclk edge depending on SIxMR[FE] mode.

Table 32-2. Interface A—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TDM[A:D]:RxD[0:3]	I	Receive data from TDM[A:D].Four bits are for nibble mode, RxD[0] used for serial-bit interface.	
		State Meaning	Asserted / Negated— according to received data.
		Timing	Assertion / Negation—Data is sampled on TDM receive clock edge, depending on SIxMR[CE] mode.
TDM[A:D]:TSYNC	I	Transmit frame sync to TDM [A:D]. For IDL circuit thus is a grant permission to transmit on the D channel.	
		State Meaning	Sync mode: SIxMR[SL] is cleared: Rising edge of TSYNC will trigger start of frame according to delay in SIxMR[TFSD]. In-frame rising edge is ignored. SIxMR[SL] is set: Falling edge of TSYNC will trigger start of frame according to delay in SIxMR[TFSD]. In-frame falling edge is ignored. Grant mode: assertion is qualified only when RSYNC (sync pulse) is set, negation is qualified regardless of RSYNC: Negated: The TDM has no grant or has lost the grant access to the ISDN D-channel. Asserted: The TDM has granted access on the ISDN D-channel
		Timing	Assertion / Negation— Sync mode: Sampled or latched on TDM Tclk edge depending on SIxMR[FE] mode. Grant mode: Assertion is qualified only when RSYNC (sync pulse) is set, negation is qualified regardless of RSYNC.
TDM[A:D]:TxD[0:3]	O	transmit data to TDM [A:D].Four bits are for nibble mode, TxD[0] used for serial-bit interface.	
		State Meaning	Asserted / Negated— according to transmitted data.
		Timing	Assertion / Negation—Data is driven on TDM transmit clock edge, depending on SIxMR[CE] mode.
TDM[A:D]:RQ	O	IDL request permission to transmit on D channel	
		State Meaning	Asserted - request to transmit data on D channel. Negated - IDLE
		Timing	Assertion / Negation—according to TDM transmit clock.

32.6 SI Register Definition

The list of all internal SI registers is shown in Table 32-3.

Table 32-3. SI Register Summary

Offset from SI1_base	Registers Name	Size	Access	Section/Page
General Registers				
0x00	SI TDM A mode register (SIAMR)	16 bits	R/W	32.6.3/32-13

Table 32-3. SI Register Summary (continued)

Offset from SI1_base	Registers Name	Size	Access	Section/Page
0x02	SI TDM B mode register (SIBMR)	16 bits	R/W	32.6.3/32-13
0x04	SI TDM C mode register (SICMR)	16 bits	R/W	32.6.3/32-13
0x06	SI TDM D mode register (SIDMR)	16 bits	R/W	32.6.3/32-13
0x08	SI global mode register high (SIGLMRH)	8 bits	R/W	32.6.2/32-13
0x0A	SI command register high (SICMDRH)	8 bits	R/W	32.6.5/32-21
0x0C	SI status register high (SISTRH)	8 bits	R	32.6.6/32-22
0x0E	SI RAM shadow address register high (SIRSRH)	16 bits	R/W	32.6.4/32-20
0x10	SI RAM counter Tx TDM A (SITARC)	8 bits	R	32.6.7/32-23
0x11	SI RAM counter Tx TDM B (SITBRC)	8 bits	R	32.6.7/32-23
0x12	SI RAM counter Tx TDM C (SITCRC)	8 bits	R	32.6.7/32-23
0x13	SI RAM counter Tx TDM D (SITDRC)	8 bits	R	32.6.7/32-23
0x14	SI RAM counter Rx TDM A (SIRARC)	8 bits	R	32.6.7/32-23
0x15	SI RAM counter Rx TDM B (SIRBRC)	8 bits	R	32.6.7/32-23
0x16	SI RAM counter Rx TDM C (SIRCRC)	8 bits	R	32.6.7/32-23
0x17	SI RAM counter Rx TDM D (SIRDRC)	8 bits	R	32.6.7/32-23
0x40	Reserved	32 bits	R/W	
0x44	SI enhanced SDM (SIENS)	8 bits	R/W	32.6.8/32-23
0x48–0x64	Reserved	256 bits	R/W	

32.6.1 SI RAM Entry

SI RAM is shown in Figure 32-8.

Access: Read/Write

	0	1	2	3	4	5	6	7	10	11	13	14	15
R W	—	SWTR	SSEL 1	SSEL 2	SSEL 3	SSEL 4	—	CSEL		CNT		BYT	LST
Reset	All zeros												

Figure 32-8. SI RAM Entry for UCC

Table 32-4 describes the SI RAM fields.

Table 32-4. SI RAM Entry Field Description

Bits	Name	Description
0	—	Reserved, should be cleared.
1	SWTR	Switch Tx and Rx. SWTR should be asserted in both Receive and 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.
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 SI RAM entries, so the value on a strobe pin is the logical OR of the Rx and Tx RAM entry strobe bit.s Multiple strobes can be asserted simultaneously. A strobe configured to be asserted in consecutive SI 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 entry 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 SI RAM entries. 1xxx Strobe 1. x1xx Strobe 2. xx1x Strobe 3. xxx1 Strobe 4.
6	-	Reserved, should be cleared.
7–10	CSEL	Channel select 0000 The bit/byte group is not supported by the QUICC Engine block. The transmit data pin is three-stated and the receive data pin is ignored. 0001 The bit/byte group is routed to ucc5. 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved. 0110 Reserved. 0111 The bit/byte group is not supported by the QUICC Engine block. 1000 Reserved. 1001 The bit/byte group is routed to ucc1. 1010 The bit/byte group is routed to ucc2. 1011 The bit/byte group is routed to ucc3. 1100 The bit/byte group is routed to ucc4. 11xx Reserved.
11–13	CNT[0-2]	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. **Note:** In nibble mode CNT is in four bits multiplication, such as CNT=1 equals 4 bits in parallel.

Table 32-4. SI RAM Entry Field Description

Bits	Name	Description
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 SI 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.

SWTR example is shown in Figure 32-9. 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 that if the transmit and receive sections of the TDM do not use a single clock source, this feature can cause erratic behavior.

This feature does not work with nibble operation.

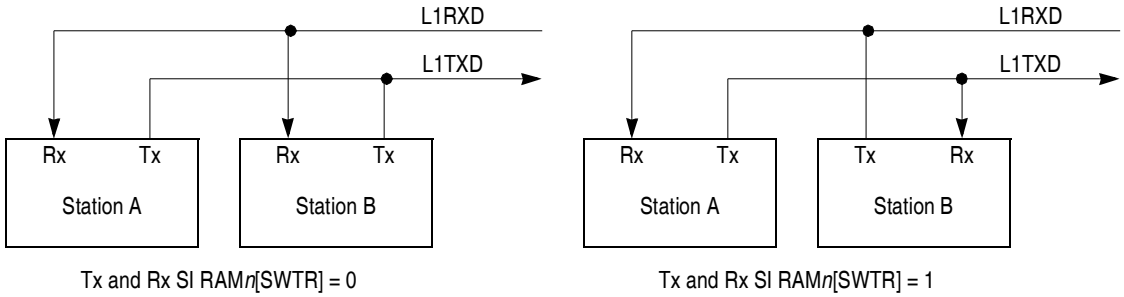


Figure 32-9. Using the SWTR Feature

32.6.2 SI Global Mode Register High (SIGLMRH)

SI Global Mode Register High is shown in Figure 32-10.

Offset 0x08 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	STZD	STZC	STZB	STZA	END	ENC	ENB	ENA
Reset	All zeros							

Figure 32-10. SI Global Mode Register High (SIGLMRH)

Table 32-5 describes SI Global Mode Register fields.

Table 32-5. SI Global Mode Register High (SIGLMRH)

Bits	Name	Description
0–3	STZx	Program L1TXD to zero for TDM A,B,C or D from reset until bit is cleared and TSYNC is detected. 0 Normal operation, high Z. 1 L1TXD = 0. This bit should be cleared by the user, and then L1TXD stays 0 until TSYNC is detected.
4–7	ENx	Enable TDMx. **Note:** Note that enabling the TDM is the last step in the initialization sequence. 0 TDM*x* is disabled. The SIx RAM and routing for TDM*x* are in a state of reset, but all other SI functions still operate. 1 All TDM*x* functions are enabled.

32.6.3 SI Mode Register (SI*x*MR)

There are four SI*x*MR registers, one for each TDM. The register is shown in Figure 32-11.

Offset SIAMR 0x00; SIBMR 0x02; SICMR 0x04; SIDMR 0x06
SIEMR 0x20; SIFMR 0x22; SIGMR 0x24; SIHMR 0x26 Access: Read/Write

	0 3	4 5	6 7	8	9	10	11	12	13	14 15
R W	SAD	SDM	RFSD	—	CRT	SL	CE	FE	GM	TFSD
Reset	All zeros									

Figure 32-11. SI Mode Register (SI*x*MR)

NOTE

Adjacent sync signals should have minimum delay between them of at least one clock cycle.

Table 32-6 describes SI Mode Register fields.

Table 32-6. SI Mode Register Description

Bits	Name	Description
0–3	SADx	Starting address for the RAM of TDM*x*. These four bits define the starting address of the SI RAM section that belongs to TDM*x* channel. The last entry of a certain TDM is determined by the LST bit in the SI RAM entry. The user must set LST within the entries of SI RAM blocks for every TDM used such as before the starting address of the next TDM. 0000 Entries 0-31, bank0 0001 Entries 32-63, bank0 0010 Entries 64-95, bank1 0011 Entries 96-127, bank1 0100 Entries 128-159, bank2 0101 Entries 160-191, bank2 0110 Entries 192-223, bank3 0111 Entries 224-255, bank3 1000 Entries 256-287, bank4 1001 Entries 288-319, bank4 1010 Entries 320-351, bank5 1011 Entries 352-383, bank5 1100 Entries 384-415, bank6 1101 Entries 416-447, bank6 1110 Entries 448-479, bank7 1111 Entries 480-511, bank7 **Note:** Each bank should be addressed by a single TDM only
4–5	SDM*x*	SI Diagnostic Mode for all four TDMs 00 normal operation. 01 Automatic echo. In this mode, the channel_x transmitter automatically retransmits the TDM received data on a bit-by-bit basis. The receive section operates normally, but the transmit section can only retransmit received data. In this mode, the L1GR*x* line is ignored. 10 Internal loopback. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXD*x* is connected to L1RXD*x*). The receiver and transmitter operate normally. The data appears on the L1TXD*x* pin and in this mode, the L1RQ*x* line is asserted normally. The L1GR*x* line is ignored. 11 Loopback control. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXD*x* is connected to L1RXD*x*). The transmitter output (L1TXD*x*) and the L1RQ*x* pin is inactive. This mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines.
6–7	RFSD*x*.	Receive frame sync delay for all four TDM. Determines the number of clock delays between the receive sync and the first bit of the receive frame. Even if CRT*x* is set, these bits do not control the delay for the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay. Use for IDL. 10 2-bit delay 01 3-bit delay
8	—	Reserved. Should be cleared.

Table 32-6. SI Mode Register Description (continued)

Bits	Name	Description
9	CRT*x*	Common receive and transmit pins for all TDM. Useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLK*x* and L1TSYNC*x* pins can be used as general-purpose I/O pins. 0 Separate pins. The receive section of this TDM uses L1RCLK*x* and L1RSYNC*x* pins for framing and the transmit section uses L1TCLK*x* and L1TSYNC*x* for framing. 1 Common pins. The receive and transmit sections of this TDM use L1RCLK*x* as clock pin of channel x and L1RSYNC*x* as the receive and transmit sync pin. Use for IDL. RFSD and TFSD are independent of one another in this mode.
10	SL*x*	Sync level for all TDM's. 0 The L1RSYNC*x* and L1TSYNC*x* signals are active on logic "1". 1 The L1RSYNC*x* and L1TSYNC*x* signals are active on logic "0".
11	CE*x*	Clock edge for all TDM's. 0 The data is transmitted on the rising edge of the clock and received on the falling edge (use for IDL). 1 The data is transmitted on the falling edge of the clock and received on the rising edge
12	FE*x*	Frame sync edge for all TDM's. Determines whether L1RSYNC*x* and L1TSYNC*x* pulses are sampled with the falling/rising edge of the channel clock. 0 Falling edge. Use for IDL. 1 Rising edge.
13	GM*x*	Grant mode for all TDM's. 0 No grant mode is supported. 1 IDL mode. A GRANT mechanism is supported if the corresponding CMXSCR[GR*x*] is set. The grant is a sample of L1GR*x* while L1TSYNC*x* is asserted. This grant mechanism implies the IDL access controls for transmission on the D channel.
14–15	TFSD*x*	Transmit frame sync delay for all TDM's. Determines the number of clock delays between the transmit sync and the first bit of the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay 10 2-bit delay 11 3-bit delay

One-clock delay from sync to data when SI*x*MR[*n*FSD] = 01 is shown in Figure 32-12.

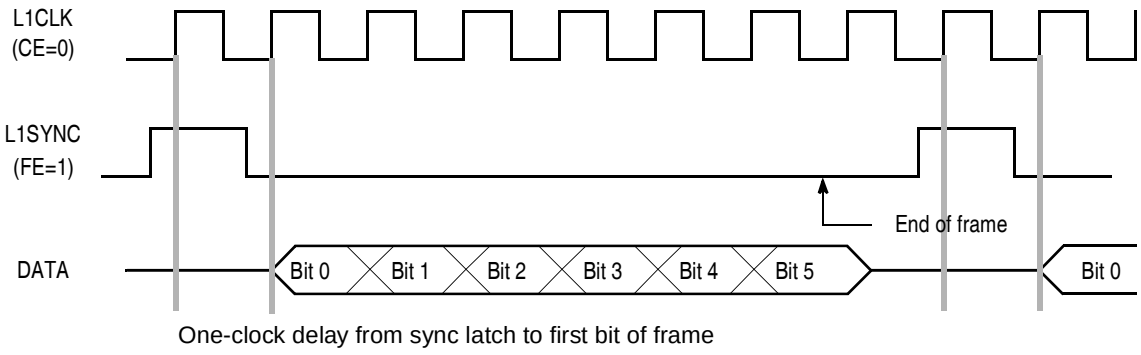


Figure 32-12. One-Clock Delay from Sync to Data (SI*x*MR[*n*FSD] = 01)

The elimination of the single-clock delay shown in Figure 32-13. Because SI*x*MR[*n*FSD] = 00 there is no delay between L1SYNC and the first bit of the frame.

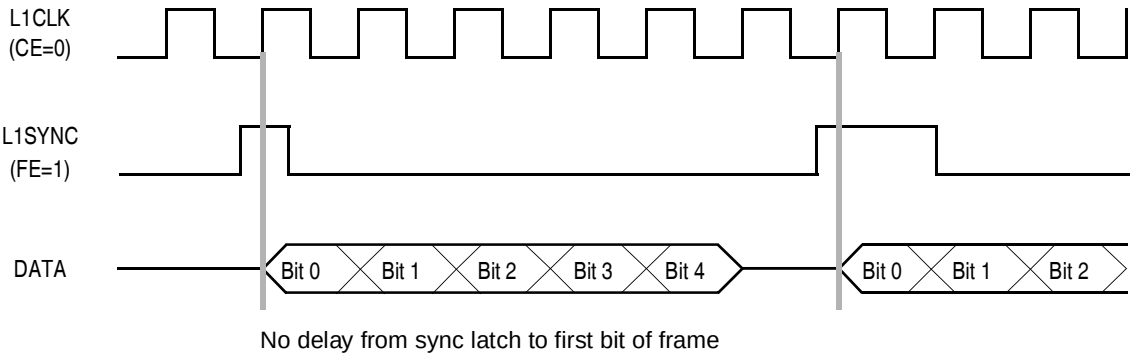


Figure 32-13. No Delay from Sync to Data (SI*x*MR[*n*FSD] = 00)

The difference between SI*x*MR[FE] = 0 and SI*x*MR[FE] = 1 is shown in Figure 32-14. In this example, SI*x*MR[CE] = 1 and there is 1-bit frame delay (SI*x*MR[*n*FSD] = 01).

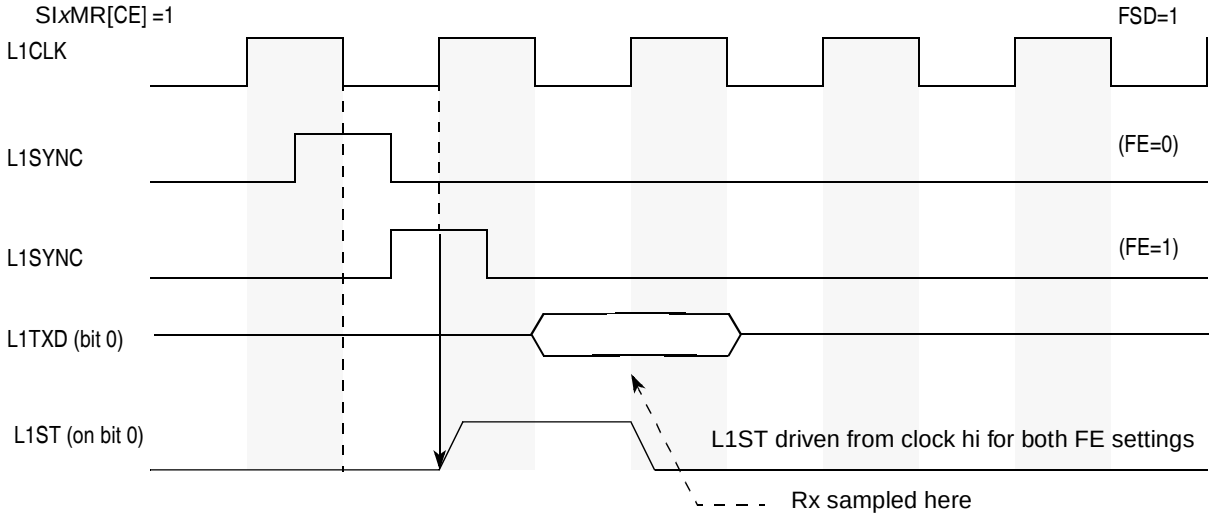


Figure 32-14. Falling Edge Effect when SI*x*MR[CE] = 1 and SI*x*MR[*n*FSD] = 01

The effect of SI*x*MR[CE] = 0 and different SI*x*MR[FE] values, with 1-bit frame sync delay is shown in Figure 32-15.

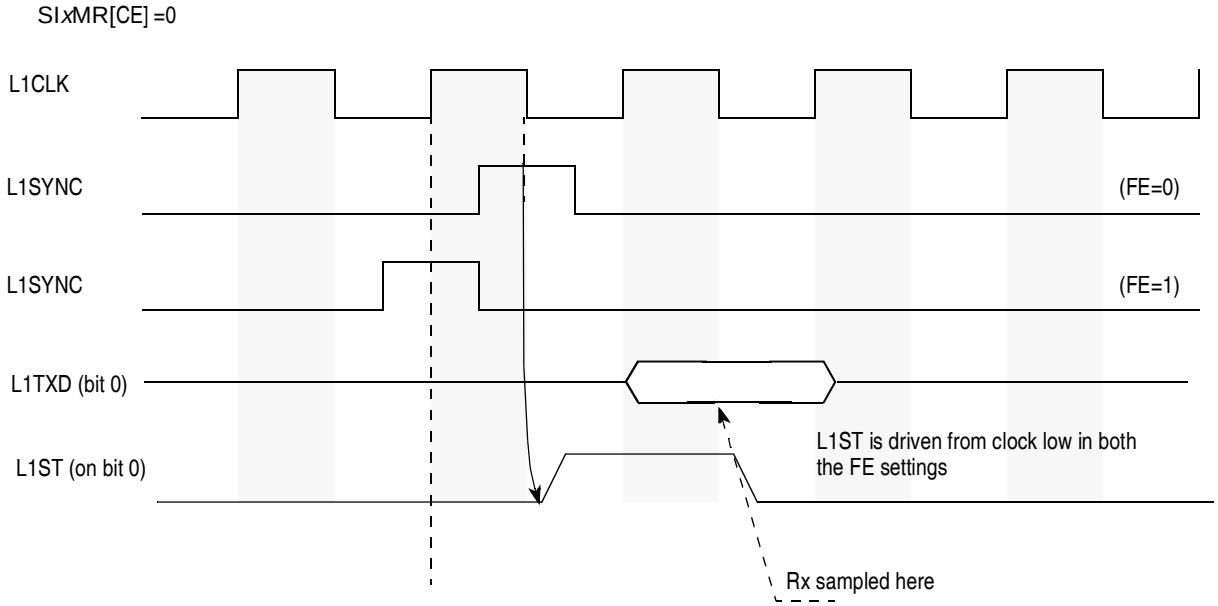


Figure 32-15. Falling Edge Effect when SI*x*MR[CE] = 0 and SI*x*MR[*n*FSD] = 01

The effect of different SI*x*MR[FE] settings when SI*x*MR[CE]=1 and no frame sync delay is shown in the following figures.

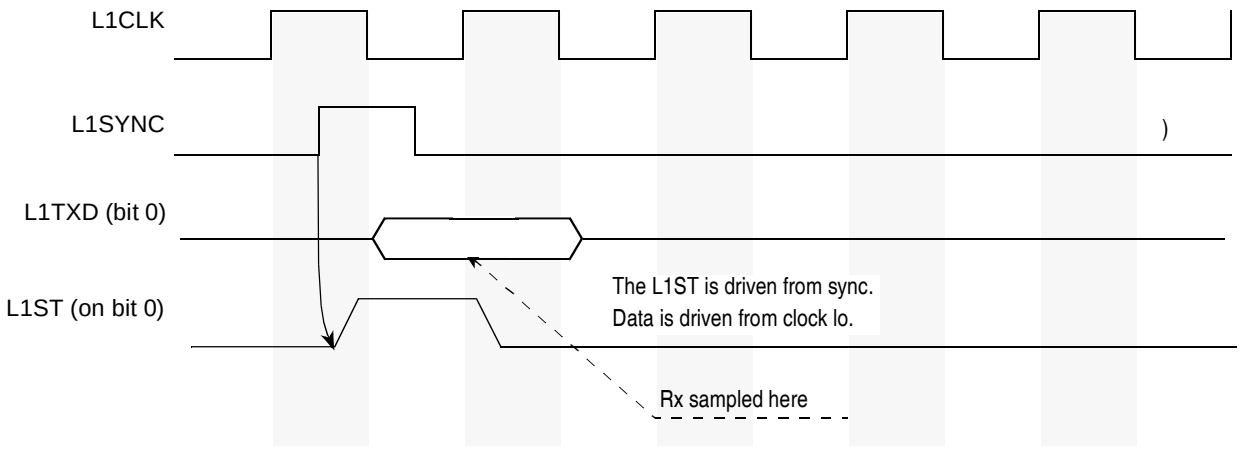


Figure 32-16. Falling Edge Effect When SI*x*MR[CE] = 1, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Falling Edge)

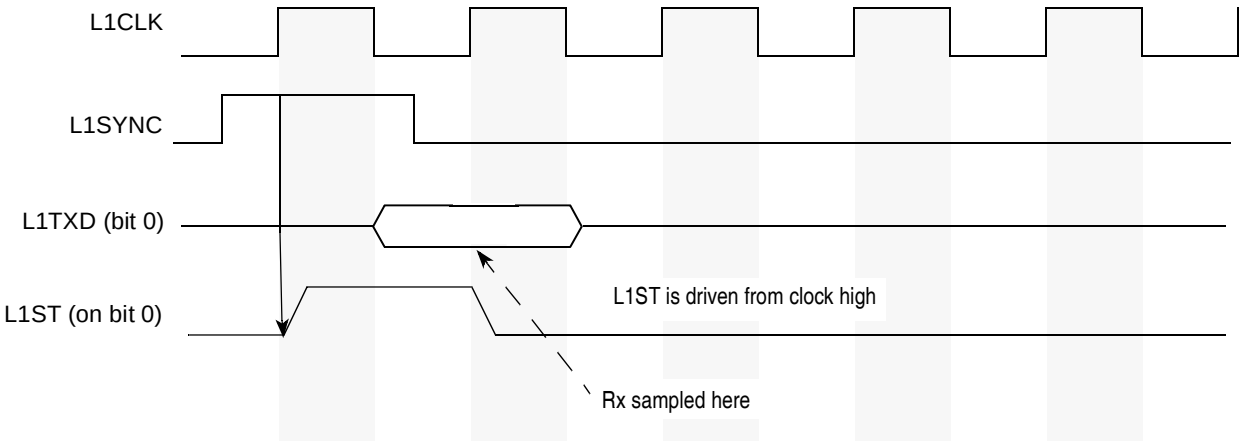


Figure 32-17. Falling Edge Effect When SI*x*MR[CE] = 1, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Rising Edge)

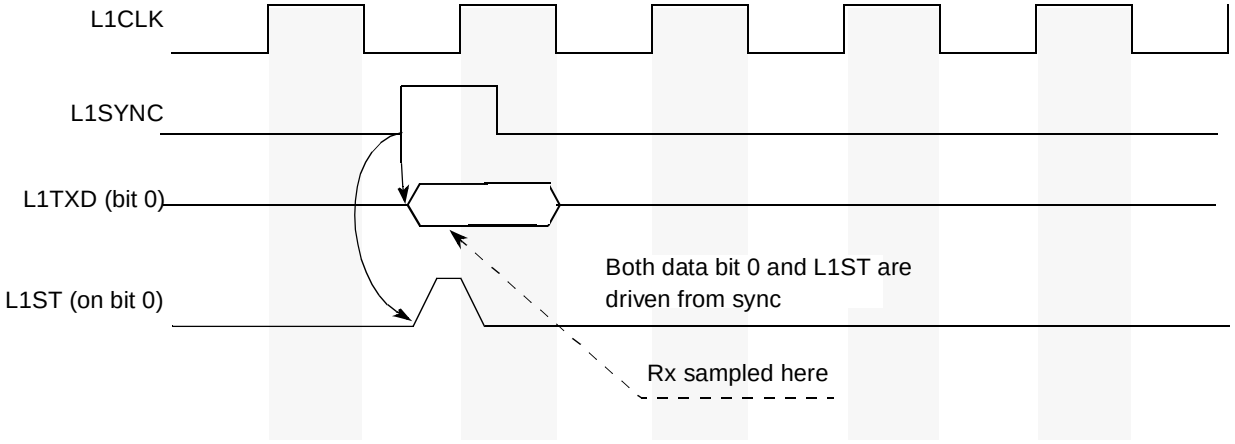


Figure 32-18. Falling Edge Effect when SI*x*MR[CE] =1, SI*x*MR[FE] =1 and SI*x*MR[*n*FSD] = 00 (Sync Starts Before Rising Edge)

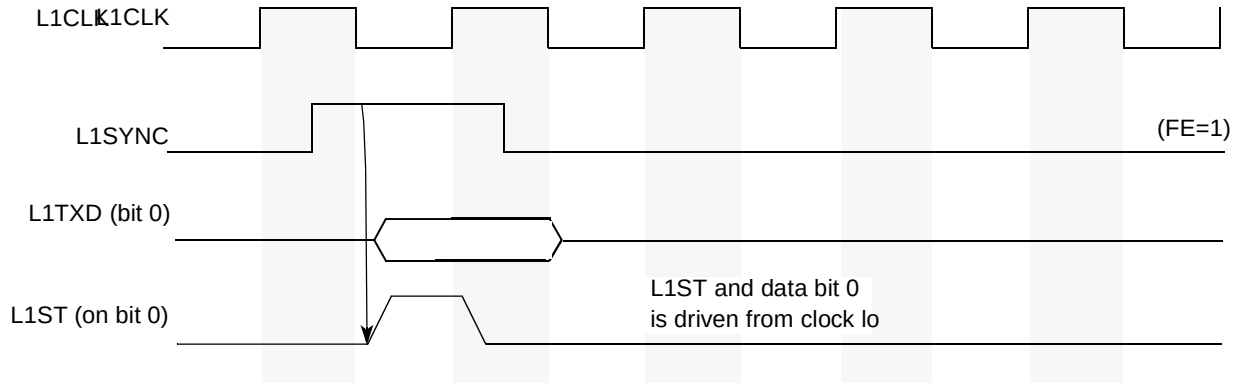


Figure 32-19. Falling Edge Effect when SI*x*MR[CE] =1, SI*x*MR[FE] =1 and SI*x*MR[*n*FSD] = 00 (Sync Starts Before Falling Edge)

The effects of different FE when SI*x*MR[CE] = 0 with no frame sync delay are shown in Figure 32-20 through Figure 32-23.

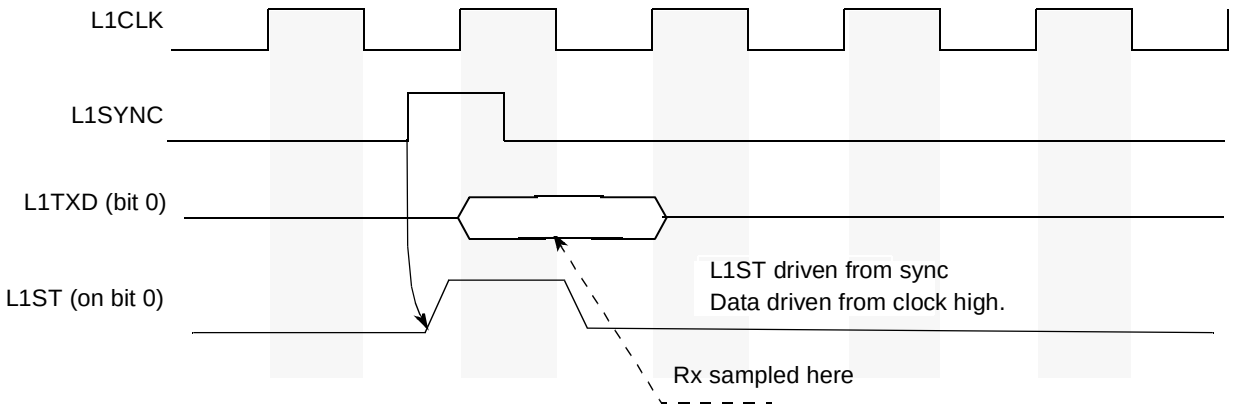


Figure 32-20. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 1 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Rising Edge)

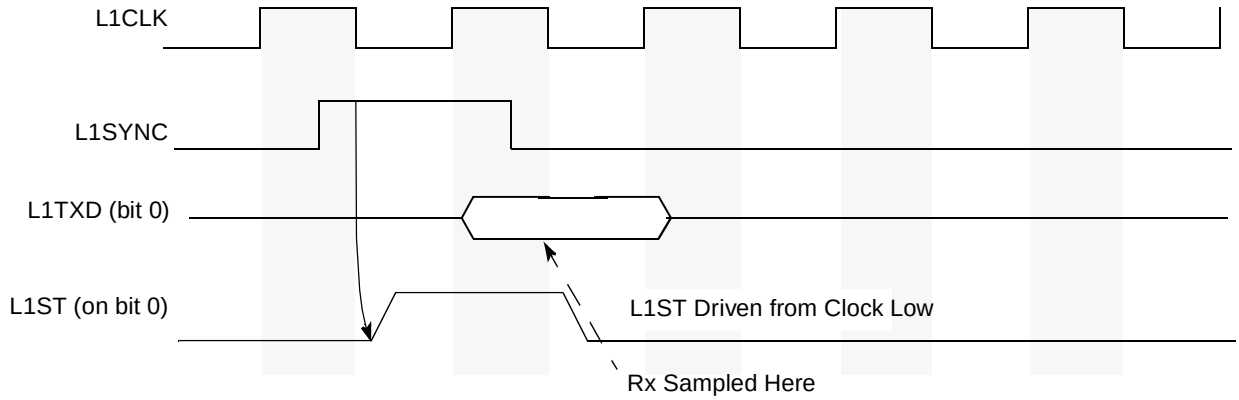


Figure 32-21. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 1 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Falling Edge)

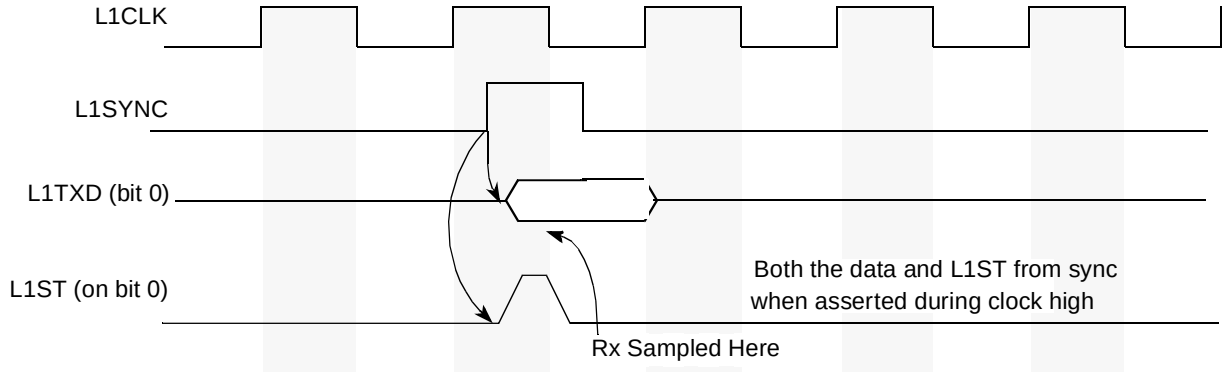


Figure 32-22. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Falling Edge)

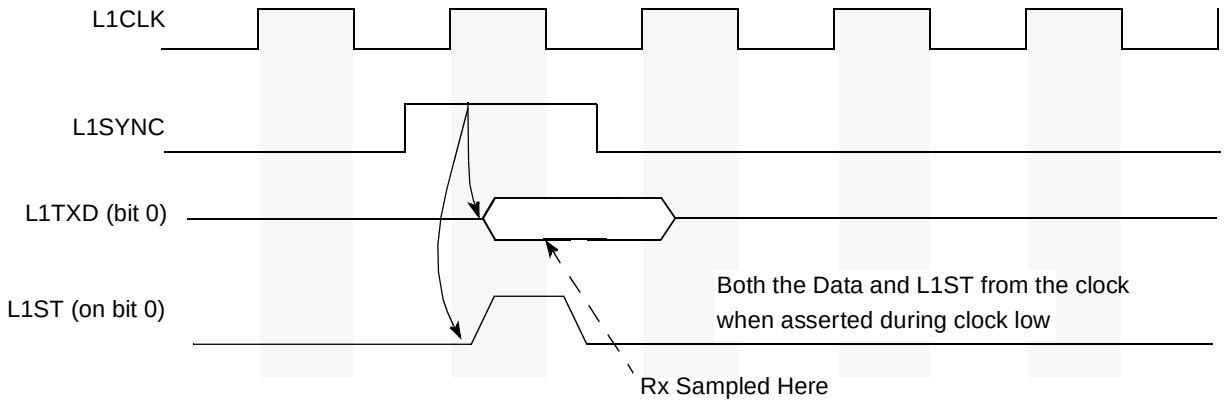


Figure 32-23. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Rising Edge)

32.6.4 SI RAM Shadow Address Register High (SIRSRH)

SIRSRH, shown in Figure 32-24, defines the starting addresses of the shadow section in the SI RAM for each of the high four TDM channels (TDM A–TDM D).

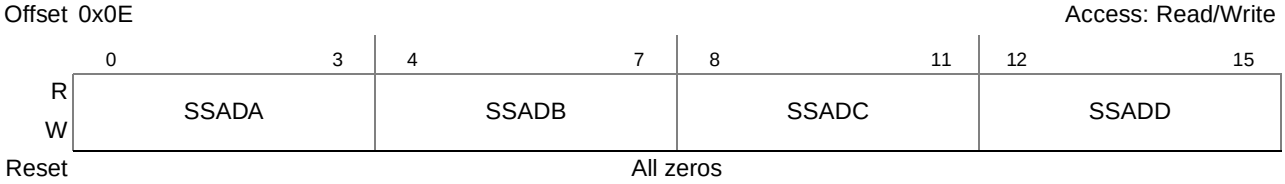


Figure 32-24. SI RAM Shadow Address Register High (SIRSRH)

SI RAM shadow address register field description is shown in Table 32-7.

Table 32-7. SI RAM Shadow Address Register High Field Descriptions

Bits	Name	Description
0–3, 4–7, 8–11, 12–15	SSAD*x*	Starting address for the RAM of TDM*x*. Defines the starting address of the SI RAM section that belongs to the TDM*x* channel. The last entry of a certain TDM is determined by the LST bit in the SI RAM entry. The user must set LST within the entries of SI RAM blocks for every TDM used— that is, before the starting address of the next TDM. 0000 Entries 0–31, bank0 0001 Entries 32–63, bank0 0010 Entries 64–95, bank1 0011 Entries 96–127, bank1 0100 Entries 128–159, bank2 0101 Entries 160–191, bank2 0110 Entries 192–223, bank3 0111 Entries 224–255, bank3 1000 Entries 256–287, bank4 1001 Entries 288–319, bank4 1010 Entries 320–351, bank5 1011 Entries 352–383, bank5 1100 Entries 384–415, bank6 1101 Entries 416–447, bank6 1110 Entries 448–479, bank7 1111 Entries 480–511, bank7 **Note:** Each bank should be addressed by a single TDM only

32.6.5 SI Command Register High (SICMDRH)

SICMDRH, shown in Figure 32-25, allows the user to program the SI RAM dynamically. When the user can set the corresponding SICMDR bit, the SI switches to the shadow RAM at the end of the current-route RAM frame.

Offset 0x0A Access: Read/Write

	0	1	2	3	4	5	6	7
R W	CSRRA	CSRTA	CSRRB	CSRTB	CSRRC	CSRTC	CSRRD	CSRTD
Reset	All zeros							

Figure 32-25. SI Command Register High

SI Command register field descriptions are shown in Table 32-8.

Table 32-8. SI Command Register High Field Descriptions

Bits	Name	Description
0,2,4,6	CSRR*x*	Change shadow RAM for all TDM's receivers. Set CSRR*x* 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 **Note:** There is a gap between setting the CSRR*x* bit by the user, and clearing it by the SI. This is because the switch can take place only after frame end.
1,3,5,7	CSRT*x*	Change shadow RAM for all TDM's transmitter. Set CRST*x* 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 transmitter shadow RAM. Cleared as soon as the switch has completed. **Note:** There is a delay between setting the CSRT*x* bit by the user, and clearing it by the SI. This is because the switch can take place only after frame end.

32.6.6 SI Status Register High (SISTRH)

SISTRH, shown in Figure 32-26, identifies the current-route RAM for high four TDM's. SISTR values are valid only when the corresponding SI*x*CMDR bit = 0.

Offset 0x0C Access: Read Only

	0	1	2	3	4	5	6	7
R W	CRORA	CROTA	CRORB	CROTB	CRORC	CROTC	CRORD	CROTD
Reset	All zeros							

Figure 32-26. SI Status Register High (SI*x*STRH)

SI status register high field descriptions are shown in Table 32-9.

Table 32-9. SI Status Register High Field Descriptions

Bits	Name	Description
0,2,4,6	CROR*x*	Current-route original receiver. Determines whether the current-route receiver RAM is the original or the shadow. 0 current-route receiver RAM is the original one. 1 The current-route receiver RAM is the shadow.
1,3,5,7	CROT*x*	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 original one. 1 The current-route transmitter RAM is the shadow.

32.6.7 SI RAM Counter (SIR*x*RC and SIT*x*RC)

There are sixteen TDM*x* RAM counters, one for each TDM, receive and transmit. They show the current RAM line number to be fetched. The register is shown in Figure 32-27.

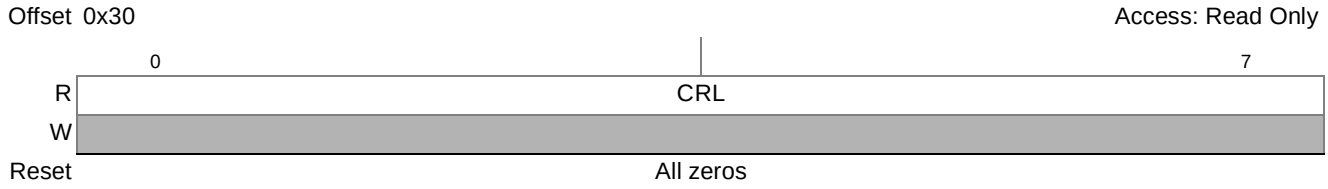


Figure 32-27. SI TDM*x* RAM Counter

Table 32-10 describes the bit fields of SIR*x*RC and SIT*x*RC.

Table 32-10. SIR*x*RC and SIT*x*RC Bit Field Descriptions

Bits	Name	Description
0–7	CRL	Current RAM line number.

32.6.8 SI Enhanced SDM Register (SIENS)

SIENS enhances diagnostic mode for UCC entries.When a TSA-related bit is set, loop/echo mode is enabled in the UCC command entry, if command entry bit1 is set. The register is shown in Figure 32-28.

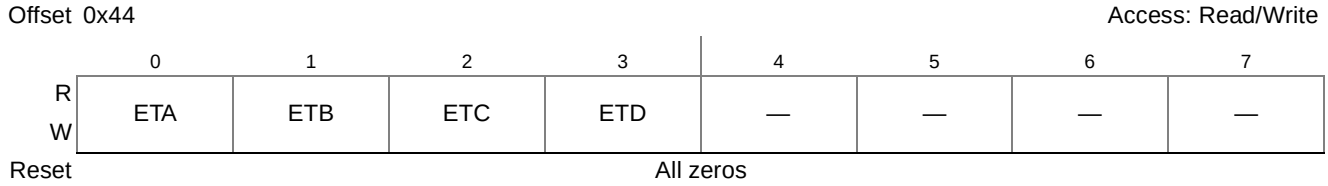


Figure 32-28. SI Enhanced SDM Register

Table 32-11 describes the SIENs bit fields.

Table 32-11. SIENS Bit Field Descriptions

Bits	Name	Description
0–7	ET*x*	0 Normal operation. 1 If bit 1 in UCC entry is set, loopback or echo mode is enabled.

32.7 SI Functional Description

32.7.1 SI RAM

The SI has a transmit RAM and a receive RAM, each with 512 entries to control TDM channel routing to all UCCs. The SI RAM entries are uninitialized after power-on reset. Unwanted results can occur if the user does not program the SI RAM before enabling the multiplexed channels.

Each line of RAM consists of two 16-bit entries that define the routing control. Each entry can control 1–8 bits or 1–32 bytes at a time as determined in the entry. 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 SI RAM can be configured in many different ways to support various TDM channels. The user can define the size of each SI RAM related to a certain TDM channel by programming the starting address of that TDM. Programming the starting shadow bank address determines whether this RAM has a shadow for changing SI RAM entries while the TDM channel is active. This reduces the number RAM entries for that TDM. Following are two examples of static and dynamic SI RAM routing.

NOTE

The switch between current SI RAM and shadow SI RAM occurs on the frame boundary.

32.7.1.1 One Multiplexed Channel with Static Frames

With this configuration, the SI RAM has 512 entries for transmit data and strobe routing and 512 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 SI RAM is determined by the user. A diagram of this configuration is shown in Figure 32-32.

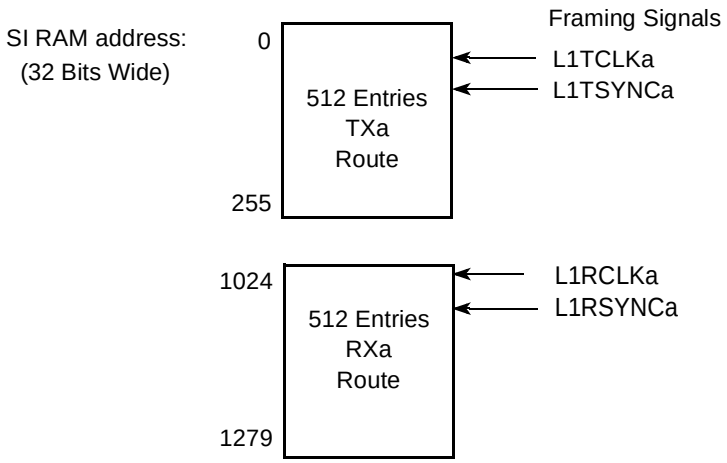


Figure 32-29. One TDM Channel with Static Frames and Independent Rx and Tx Routes

32.7.1.2 One Multiplexed Channel with Dynamic Frames

A configuration of one multiplexed channel with 256 entries for transmit and 256 entries for receive data and strobe routing is shown in Figure 32-30. 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 SICMDR[CSR*xx*] for the associated channel. When the next frame sync arrives, the SI automatically exchanges the current-route RAM for the shadow RAM.

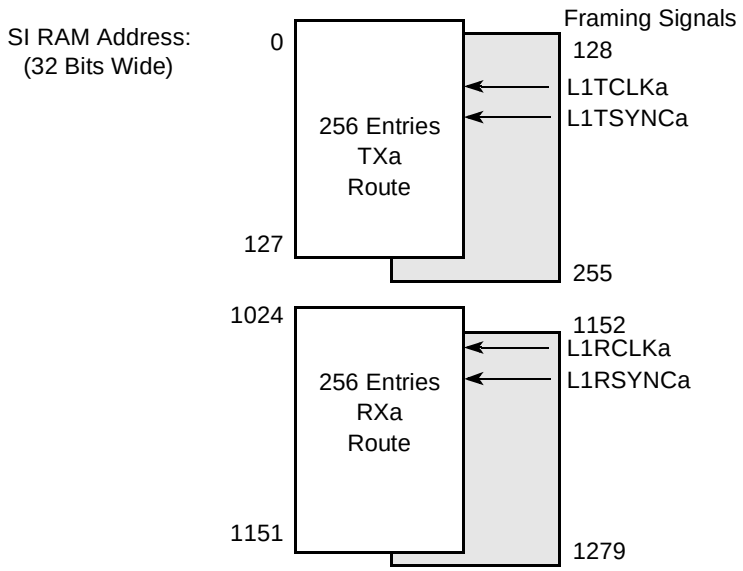


Figure 32-30. One TDM Channel with Shadow RAM for Dynamic Route Change

This configuration should be chosen when only one TDM is needed, but dynamic rerouting may be needed on that TDM. Similarly, for two TDM channels, the number of SI RAM entries are reduced for every TDM channel programmed for shadow mode.

32.7.1.3 Static and Dynamic Routing

The SI RAM has two operating modes for all TDMs:

- Static routing. The number of SI RAM entries is determined by the starting addresses the user relates to the corresponding TDM and is divided into two parts (Rx and Tx). Three requirements must be met before the new routing takes effect.
  - 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 routing definition can be modified while UCCs are connected to the TDM. A TDM channel cannot be changed dynamically if an UCC is connected to that TDM. The number of SI RAM entries is determined by the starting address 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 SI 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 SICMDR[CSR*xx*] must be set. As a result, the SI exchanges the shadow RAM and the current-route RAM as soon as the corresponding sync arrives and resets SICMDR[CSR*xx*] to signify 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. An example of the shadow RAM exchange process for two TDM channels both with half the RAM as a shadow, is shown in Figure 32-31

For example, if one TDM with dynamic changes is programmed to own all four banks and the shadow is programed to the last tow banks, the initial current-route RAM addresses in the SI RAM are as follows.

- 0–255: TXa route
- 1024–1279: RXa route

The 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 do not attempt to write the current-route RAM. The SI status register (SISTR) 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 SI RAM entry starting or ending execution by the TSA.

An example is shown in Figure 32-31.

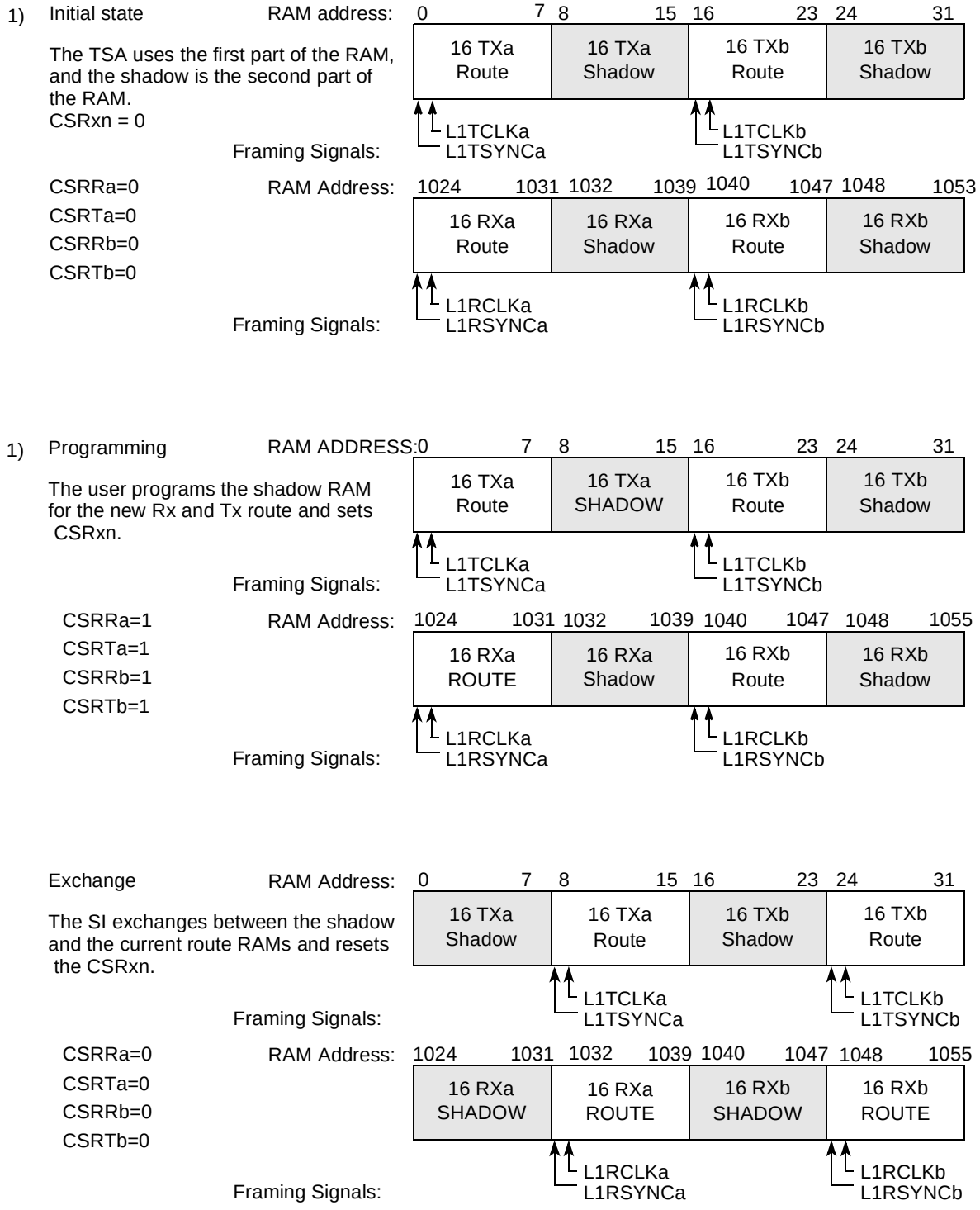


Figure 32-31. Example: SI RAM Dynamic Changes, TDM A and TDM B, Same SI RAM Size

32.7.2 Time-Slot Assigner

The TSA has two separate TSA inside: Rx TSA and Tx TSA.

32.7.2.1 Tx TSA

The Tx TSA implements route selection of data transmitted from the UCCs according to the relevant TDM SI RAM entries. All its internal registers are uninitialized after power-on reset.

After the TSA receives the SYNC signal from a TDM, data is transferred from the UCC to the TDM This procedure occurs for all TDMs.

32.7.2.2 Rx TSA

The Tx TSA implements route selection of data received from the TDMs to UCCs according to the relevant TDM SI RAM entries.All internal registers are uninitialized after power-on reset.

After the TSA receives the SYNC signal from the TDM, data is transferred from the TDM to the UCC. This procedure occurs for all TDMs.

32.8 SI Initialization Information

Following are steps to initialize the SI for proper operation:

1. Initialize the relevant SIxRAM entries.
2. Initialize the SI mode register (SI*x*MR).
3. Connect the UCCs to SI using CE_MUX.
4. Connect the clocks to the SI using CE_MUX.
5. Configure the TDMs, UCCs.
6. Enable the TDMs in the SI global mode register, SI*x*GMR.

NOTE

Changing the SI register in a different order; that is, changing SIxMR in the middle of a frame, can cause erratic behavior.

32.9 SI Application Information

32.9.1 SI RAM Programming Example

The example in this section shows how to program the RAM to support the 10-bit IDL bus shown in Figure 32-34. The TSA supports the B1 channel with UCC2, the D channel with UCC1, 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 UCC3. 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)—UCC2

- 1 bit (D)—UCC1 + strobe1
- 1 bit—no support
- 4 bits (B2)—strobe2
- 4 bits (B2)—UCC3
- 1 bit (D)—UCC1 + strobe1

Each of these six divisions can be supported by a single SI RAM entry. Thus, six SI RAM entries are needed. See Table 32-12.

Table 32-12. RAM Word Descriptions

Entry Number	RAM Word						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
1	0	0000	1010	000	1	0	8-bit UCC2
2	0	1000	1001	000	0	0	1-bit UCC1 strobe1
3	0	0000	0000	000	0	0	1-bit no support
4	0	0100	0000	011	0	0	4-bit strobe2
5	0	0000	1011	011	0	0	4-bit UCC3
6	0	1000	0001	000	0	1	1-bit UCC1 strobe1

Note that because IDL requires the same routing for both receive and transmit, an exact duplicate of these entries should be written to both the receive and transmit sections of the SI RAM. Then SIxMR[CRTx] can be used to instruct the SI RAM to use the same clock and sync to control both sets of SI RAM entries simultaneously.

32.9.2 One Multiplexed Channel with Static Frames

In the example configuration shown in Figure 32-32, the SI RAM has 512 entries for transmit and 512 entries for data and strobe routing. This configuration should be chosen when only one TDM is required and its routing does not need to be changed dynamically. The number of entries available in the SI RAM is determined by the user.

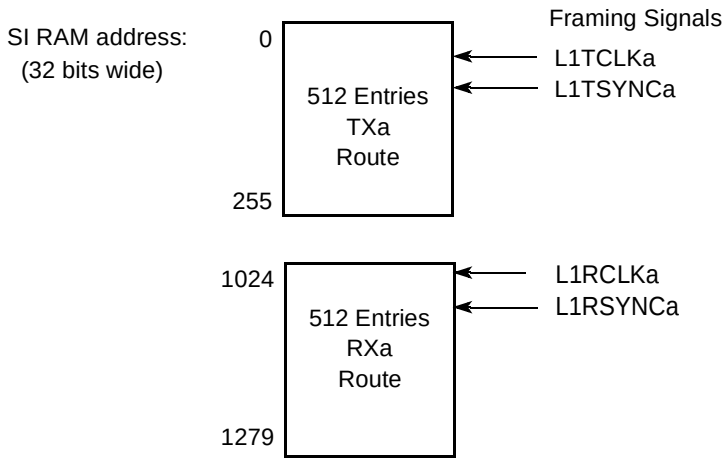


Figure 32-32. One TDM Channel with Static Frames and Independent Rx and Tx Routes

32.9.3 One Multiplexed Channel with Dynamic Frames

A configuration of one multiplexed channel, shown in Figure 32-30, has 256 entries for transmit and 256 entries for receive data and strobe routing. The IDL interface is a full-duplex ISDN interface to connect a physical layer device to the QUICC Engine block, which 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 160-kbps full-duplex bandwidth. The QUICC Engine block is an IDL slave device that is clocked by the IDL bus master (physical layer device) and has separate receive and transmit sections. Because the QUICC Engine block can support all TDMs, it can actually support four independent IDL buses (the limitation is due to the number of serials that support IDL) using separate clocks and sync pulses (for two IDL buses) as illustrated in Figure 32-33.

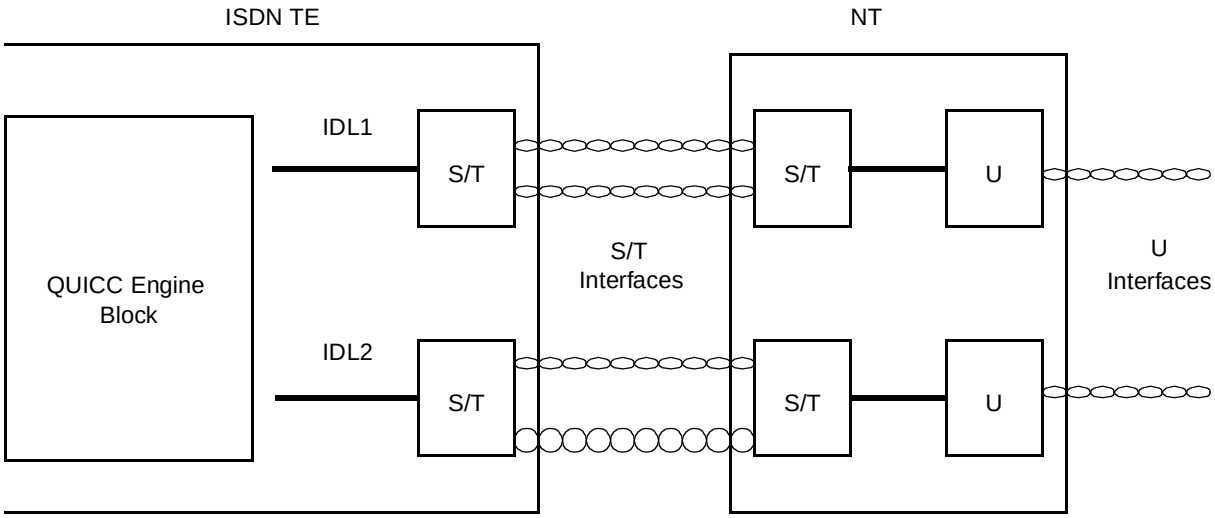


Figure 32-33. Dual IDL Bus Application Example

32.9.4 IDL Interface Example

An example of the IDL application is the ISDN terminal adaptor illustrated in Figure 32-34. The IDL interface connects the 2B+D channels between the QUICC Engine block, CODEC, and S/T transceiver. One of the QUICC Engine UCCs is configured to HDLC mode to handle the D channel; another QUICC Engine block UCC rate adapts the terminal data stream over the first B channel. That UCC 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 sends initialization commands and periodically checks status from the S/T transceiver. Another UCC connected to the terminal is configured for UART. The QUICC Engine block can identify and support each IDL channel or 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 32-13.

Table 32-13. IDL Signal Descriptions

Signal	Description
L1RCLKx	IDL clock; input to the device.
L1RSYNCx	IDL sync signal; input to the device. This signal indicates that the clock periods following the pulse designate the IDL frame.
L1RXDx	IDL receive data; input to the device. Valid only for the bits supported by the IDL; ignored for any other signals present.
L1TXDx	IDL transmit data; output from the device. Valid only for the bits that are supported by the IDL; otherwise, three-stated.
L1RQx	IDL request permission to transmit on the D channel; output from the device on the L1RQx pin.
L1GRx	IDL grant permission to transmit on the D Channel; input to the device on the L1TSYNCx pin.

NOTE

x = A:D for all TDMs.

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 32-34.

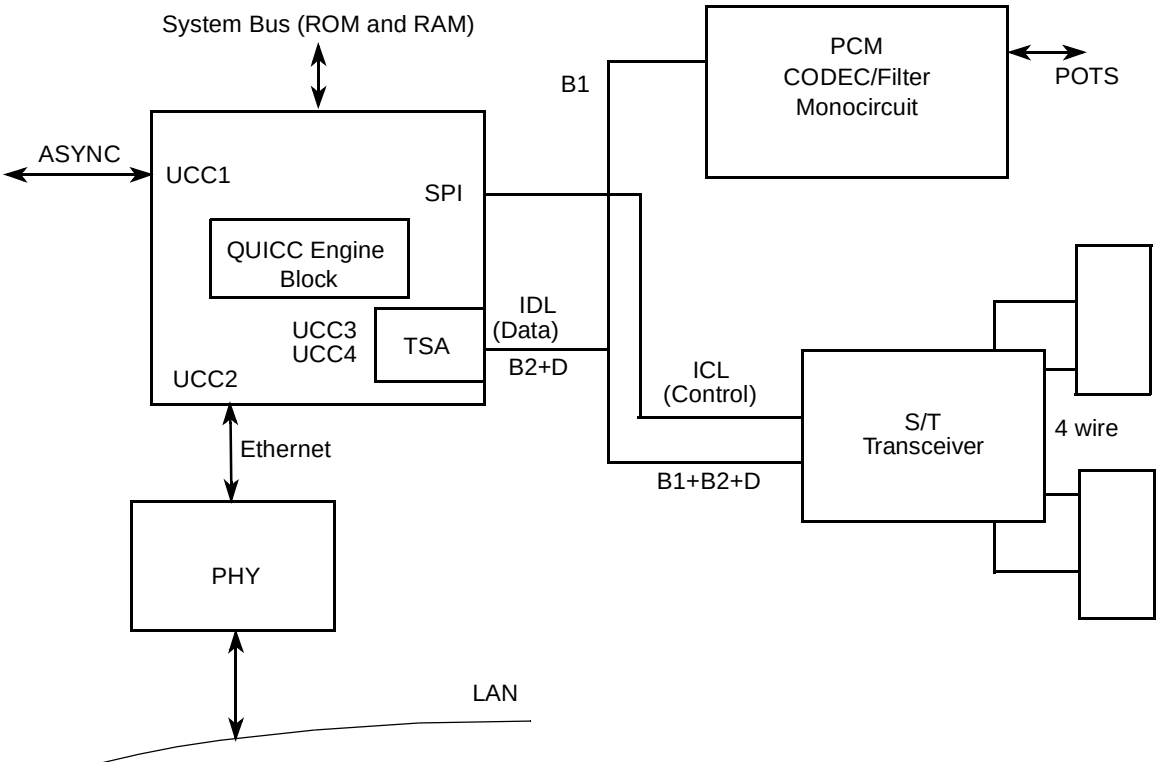
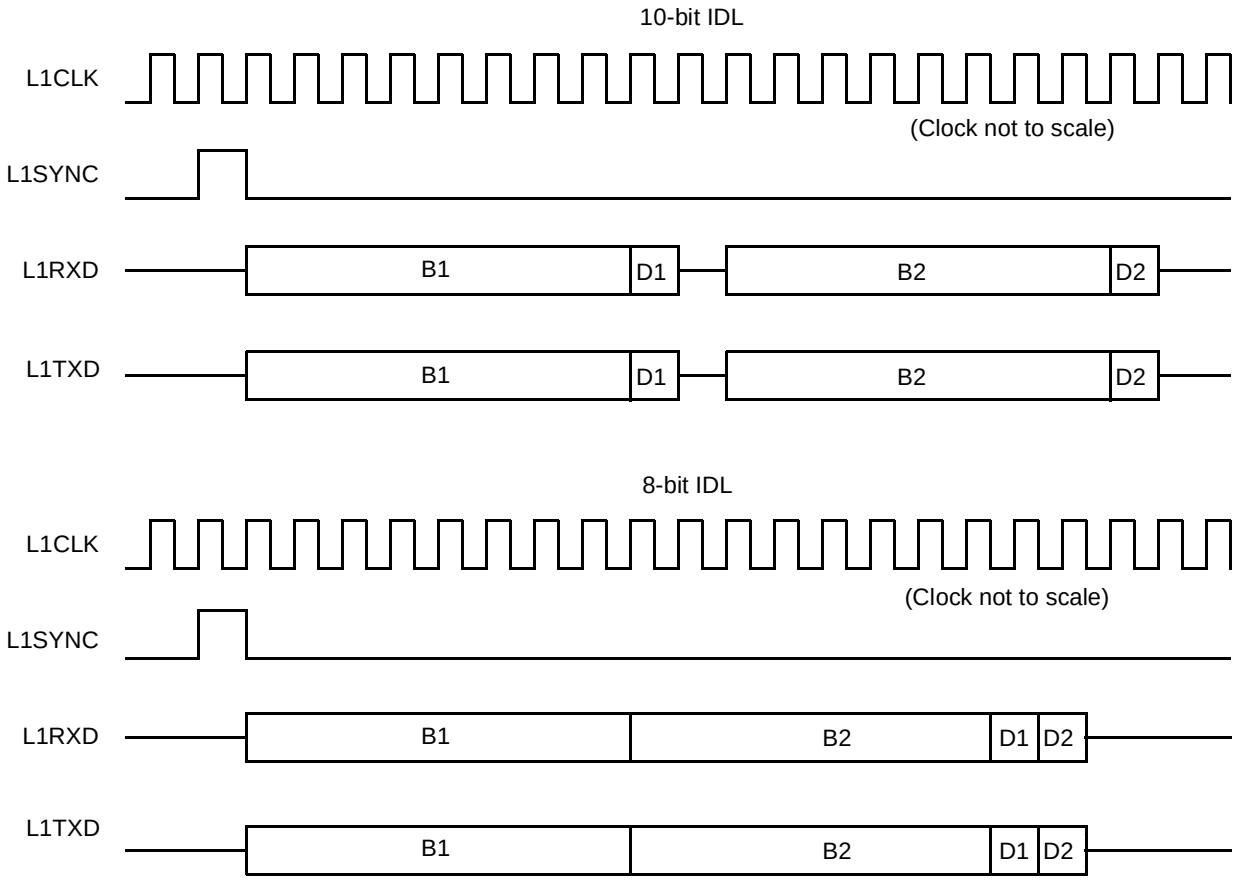


Figure 32-34. IDL Terminal Adaptor



NOTE: L1RQn and L1GRn are not shown.

Figure 32-35. IDL Bus Signals

NOTE

Previous versions of Freescale IDL-defined bit functions, called auxiliary (A) and maintenance (M), were eliminated from the IDL definition when the IDL control channel became out-of-band. They were defined as a subset of the Freescale SPI format called serial control port (SCP). If a user prefers to implement the A and M bit functions as originally defined, the TSA can be programmed to access these bits and route them transparently to a UCC. The QUICC Engine block SPI can perform this out-of-band signaling.

The QUICC Engine block supports all channels of the IDL bus in the basic rate. Each bit in the IDL frame can be routed to every UCC (connected to the specific SI) or can assert a strobe output for supporting an external device. The QUICC Engine block supports the request-grant method for contention detection on the D channel of the IDL basic rate. When the QUICC Engine block has data to transmit on the D channel, it asserts L1RQx. The physical layer device monitors the physical layer bus for activity on the D channel and indicates that the channel is free by asserting L1GRx. The QUICC Engine block samples the L1GRx signal when the IDL sync signal (L1RSYNCx) is asserted. If L1GRx is high (active), the QUICC Engine block transmits 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 L1GRx. The QUICC Engine block then stops sending

and retransmits the frame when L1GRx is reasserted. This procedure is handled automatically for the first two buffers of a frame.

For the primary rate IDL, the QUICC Engine block supports up to four 8-bit channels in the frame, determined by the SI RAM programming. To support more channels, the user can route more than one channel to every UCC and the UCC will treat it as one high-speed stream and store it in the same data buffers (this approach is appropriate only for transparent data). Additionally, the QUICC Engine block 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 L1TXDx.

32.9.5 IDL Interface Programming

The user can program the channels for the IDL bus interface to the appropriate configuration. First, the SIxMR should be programmed to the IDL grant mode for that channel through the GMx bits. More than one channel can be programmed to interface with the IDL bus. If receive and transmit sections are used for interfacing to the same IDL bus, the user can internally connect the receive clock and sync signals to the SI RAM transmit section, using the CRTx bits. The RAM section for the IDL channels must be programmed to the preferred routing. See Section 32.9.1, “SI RAM Programming Example.”

Next, define the IDL frame structure to be a delay of 1-bit from frame sync to data, to falling edge sample sync, and the clock edge to transmit on the rising edge of the clock. The L1TXDx pin should be programmed to be three-stated when inactive (through the parallel I/O open-drain register). To support the D channel, the user must program the appropriate CMXS1CR[GRx] bit and program the RAM entry to route data to that serial controller. The two definitions of IDL, 8 and 10 bits, are supported only by modifying the SI RAM programming. In both cases, L1GRx is sampled with the L1TSYNCx signal and transferred to the D channel UCC as a grant indication. The same procedure is valid for supporting an IDL bus in the second channel.

Chapter 33
Serial ATM Microcode

The serial ATM microcode (SAM) complies with the ITU-T I.432 (transmission convergence sublayer) for SDH-based ATM systems. From the perspective of the QUICC Engine block, the SAM operates on ATM cells transferred over the SI through the UCC. Figure 33-1 shows the structure of the SAM.

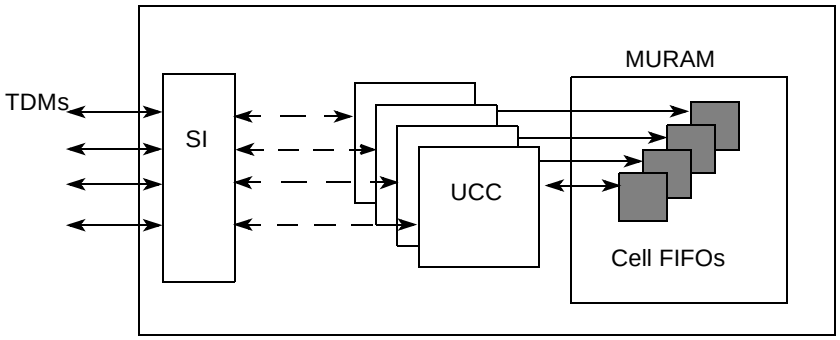


Figure 33-1. Serial ATM Microcode Structure

Each UCC controls one TDM link and therefore one TC layer. The routing of cells to the UCC TC layer is programmed in the SIRAM.

33.1 Features

The main features of the SAM include the following:

- Operates on UCC
- Four TDM channels routed by the SI to 4 Microcode TC layer blocks
  - — Protocol-specific overhead bits may be discarded or routed to other controllers by the SI
  - — Performing ATM TC layer functions (according to ITU-T I.432)
  - — Transmit (Tx) SAM features are as follows:
    - – Cell HEC generation
    - – Payload scrambling using self synchronizing scrambler (programmable by the user)
    - – Coset generation (programmable by the user)
    - – Cell rate decoupling by inserting idle cells
    - – Programmable cell FIFO size
  - — Receive (Rx) SAM features are as follows:
    - – Cell delineation using bit by bit HEC checking and programmable ALPHA and DELTA parameters for the delineation state machine

  - Bit and byte synchronous cell delineation modes
  - Payload descrambling using self synchronizing scrambler (programmable by the user)
  - Coset removing (programmable by the user)
  - Filtering idle/unassigned cells (programmable by the user)
  - Performing HEC error detection and single bit error correction (programmable by the user)
  - Generating loss of cell delineation status/interrupt (LOC/LCD)
  - Programmable cell FIFO size
- Supports the TC layer and PMD (physical medium dependent) WIRE interface (according to the ATM-Forum af-phy-0063.000)
- Statistical data per TC layer via cell counters

33.2 Microcode TC Layer (MTC)

The channels present within TDM frame are connected to an MTC, the MTC is the entity responsible for performing the ITU-T I.432 specific tasks. It is also responsible for interfacing to the SI through the UCC and provides the bridge from the ATM/IMA microcodes to the physical transmission medium. The MTC relationship to other QUICC Engine module resources is depicted in Figure 33-2.

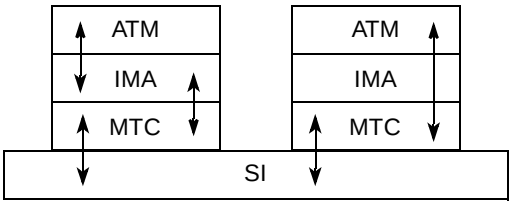


Figure 33-2. QUICC Engine ATM Stack with MTC

On a per-MTC basis, the MTC can be configured to pass cells to/from the IMA microcode or directly to/from the ATM controller of the QUICC Engine block.

33.2.1 Transmit MTC

The transmit MTC architecture is illustrated in Figure 33-3.

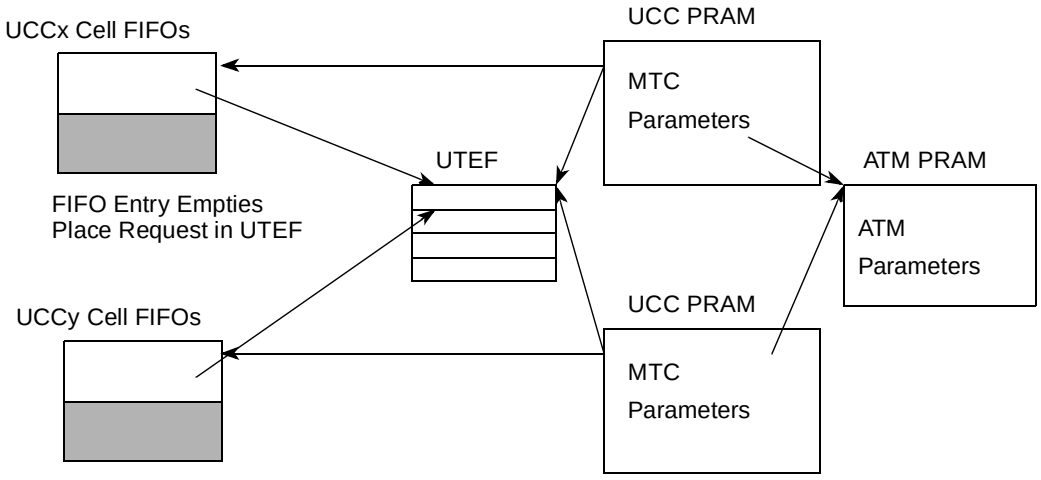


Figure 33-3. MTC Transmit Architecture

Each MURAM-based cell FIFO ring is unique to the MTC controlling the ring. The size of the Tx cell FIFO ring is programmable.

When a cell FIFO entry is emptied, a request is made to the ATM controller for a cell to be transferred to the FIFO, and the UCC moves to the next cell FIFO entry in the ring to begin transmitting the next cell. The request is placed into the UTOPIA emulation FIFO (UTEF). The ATM microcode polls the UTEF, and when an entry is added this triggers the ATM/IMA controllers in the QUICC Engine block to deliver a cell to the emptied FIFO entry. As Figure 33-3 illustrates, more than one MTC can be directed to the same UTEF. This configuration is analogous to a multi-PHY configuration of the UTOPIA bus. Multiple MTCs place requests into the same UTEF, and each MTC routes the received cells to the ATM layer through the same ATM Parameter RAM. Each MTC has its own PHY identifier that the ATM controller interprets as a UTOPIA PHY address. Requests from different MTCs to a shared UTEF are handled in the order in which they are added to the UTEF. This emulates UTOPIA polling. A faster PHY has more requests added to the UTEF, which is analogous to more Clav assertions.

The ATM parameter RAM (and hence the ATM controller) must be that of UCC1–5.

33.2.1.1 UCC Transmitter

When a UCC controls an MTC, the UCC Parameter RAM should be used to hold the MTC parameters. The UCC begins transmitting the cell by accessing the current entry in the cell FIFO. It transmits four bytes at a time, and the MTC passes it four bytes of the cell payload from current cell FIFO entry—except when the HEC byte is passed to the UCC and the MTC passes only one byte to the UCC. After the MTC has passed all the cell payload to the UCC for transmission, the current cell FIFO entry is emptied and the cell FIFO extract pointer (MTC_TX_CF_XP) is incremented. The MTC then starts to pass the cell to the UCC from the entry to which the incremented MTC_TX_CF_XP is pointing. Meanwhile, the MTC requests that a cell be delivered from the ATM or IMA microcode to the emptied cell FIFO entry. This cell must be

delivered before the UCC has transmitted the cell present at the incremented MTC_TX_CF_XP, or an underrun condition occurs.

The UCC that runs the SAM microcode must be programmed to operate as a slow communications controller and must be programmed for QMC mode in the GUMR_L.

33.2.2 Receive MTC

The receive MTC architecture is illustrated in Figure 33-4.

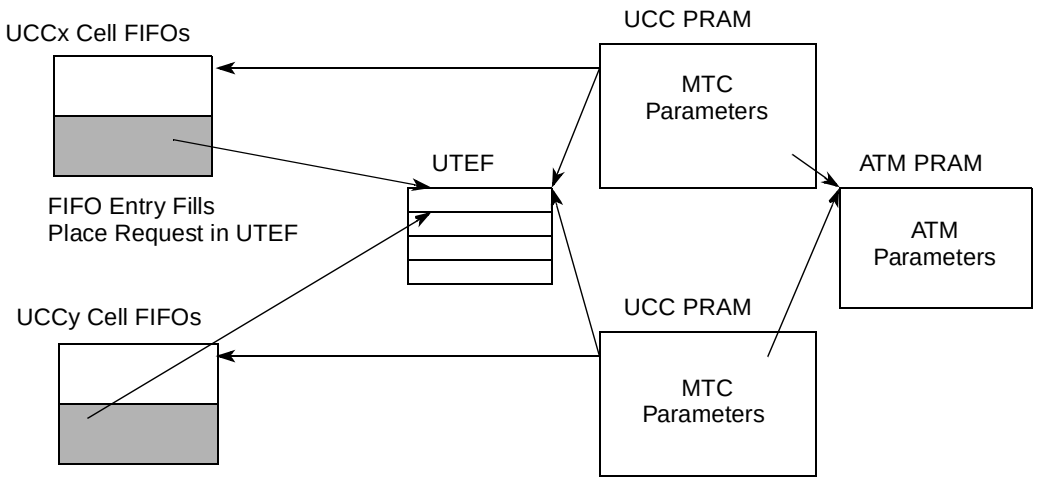


Figure 33-4. MTC Receive Architecture

The MTC receive architecture is similar to that of the transmitter. Each MTC has its own programmable cell FIFO receive ring. The ATM microcode polls the receiver UTEF. When a cell is received into the cell FIFO, a request placed into the UTEF, triggers the CPM ATM controller to remove the cell from the FIFO and pass the cell to the ATM or IMA layers. Multiple receive MTCs can use the same UTEF so that multiple MTCs can be behave like a multi-PHY configuration on the UTOPIA bus. One ATM controller processes cells received from multiple TC layers through a single ATM controller's PRAM. Of course, each MTC can have its own unique non-shared PRAM and therefore its own UTEF.

The ATM PRAM must be that of UCC1-5, so the SNUM used for the ATM controller must be UCC1-5.

33.2.2.1 UCC Receiver

When a UCC controls an MTC, the UCC PRAM should be used to hold the MTC parameters. Each UCC channel works on 4 bytes at a time and these 4 bytes are passed to the MTC. The MTC performs the ITU-T I.432 cell delineation on the 4 bytes by verifying whether a valid HEC is found. When the receiver is synchronized, the MTC uses the first 4 bytes of the cell after SYNC state is reached to start building a cell in the first cell FIFO entry. The cell is then built 4 bytes at a time. When a full cell is received, MTC_RX_CF_FP is incremented and a request is placed in the UTEF. The next 4 bytes passed to the MTC from the UCC are used to start building a new cell in the FIFO ring. The ATM microcode processes the fully received cell and then releases this cell FIFO entry when cell processing is complete by incrementing MTC_RX_CF_XP. Cell processing must complete before the UCC receives another cell in entirety or an overrun condition occurs.

The UCC that runs the SAM microcode must be programmed to operate as a slow communications controller and must be programmed for QMC mode in the GUMR_L.

33.2.3 I.432 Functionality

During transmission of a cell, the MTC performs cell header HEC generation, a coset on HEC through a programmable coset field, and scrambling using generator polynomial $x^{43} + 1$. The HEC is generated using the polynomial $x^8 + x^2 + x + 1$. The coset polynomial $x^6 + x^4 + x^2 + 1$ is added (modulo 2) (programmable by the user) to the calculated HEC octet. When the transmit FIFO is empty, the MTC inserts idle cells (counted in ICC) and optionally generates an underrun interrupt to the core. The MTC transmit function also includes a transmit cell counter (TCC).

The MTC receive functions include cell delineation, cell payload descrambling, HEC verification, and correction and idle/unassigned cell filtering. For cell delineation the MTC searches the incoming bit stream in two modes, bit synchronous or byte synchronous. In bit synchronous mode the MTC looks at every 33rd bit received and checks for a valid HEC. In byte synchronous mode the MTC checks every byte for a valid HEC. While searching for the cell boundary, the cell delineation state machine is in HUNT state. When a correct HEC is found, the MTC locks on the particular cell boundary and enters the PRESYNC state, which indicates that the previously detected HEC pattern is not a false indication. If a correct HEC pattern is false, an incorrect HEC is received within the next DELTA cells. If an incorrect cell is detected, then a transition to the HUNT state is made. If an incorrect HEC is not detected in the PRESYNC state, a transition to the SYNC state is made after DELTA cells. In the SYNC state, the MTC is assumed to be synchronized so that other functions can be applied to the received cell. A transition back to the HUNT state is made only after ALPHA consecutive incorrect HEC patterns are detected. When the cell delineation enters or leaves the SYNC state an interrupt can optionally be generated to the core.

The cell delineation state machine is shown in Figure 33-5. The ALPHA and DELTA parameters determine the robustness of the delineation method. ALPHA determines the robustness against false alignment due to bit errors. DELTA determines the robustness against false delineation in the synchronization. Both parameters are programmable for each TC block and are provided to help tune the system according to the line error characteristics of a specific application.

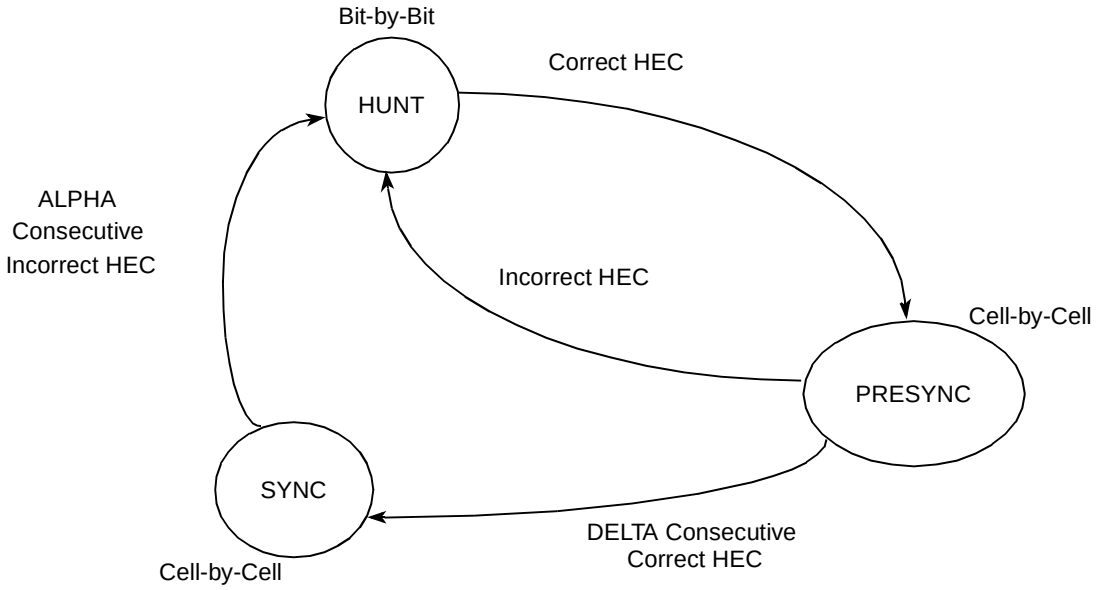


Figure 33-5. TC Cell Delineation State Machine

The MTC descrambles (programmable) the cell payload using the self-synchronizing descrambler with a polynomial of $x^{43} + 1$.

The HEC calculation is a CRC-8 calculation over the first four octets of the ATM cell header. The MTC verifies the received HEC using the accumulation polynomial, $x^8 + x^2 + x + 1$. The coset polynomial $x^6 + x^4 + x^2 + 1$ is added (modulo 2) to the received HEC octet before comparison with the calculated result (programmable).

The MTC can perform single bit error correction on the header. If multiple bit errors are found in the HEC, the cell is discarded. If the single bit error correction mode is not enabled (MTC_MODE[SBC] = 1), the cell is also discarded when a single bit error is found in the header. The MTC references a correction table in the multi-user RAM (MURAM) in order to determine if correction is possible.

When the cell delineation state machine is in the SYNC state, the HEC verification state machine (see Figure 33-6) implements the correction algorithm. This state machine ensures that a single cell header is corrected at a time. If consecutive cells are detected with single bit errors in their headers, only the first cell error is corrected and the rest are discarded. This state machine reduces the probability of the delivery of cells with erroneous headers under bursty error conditions.

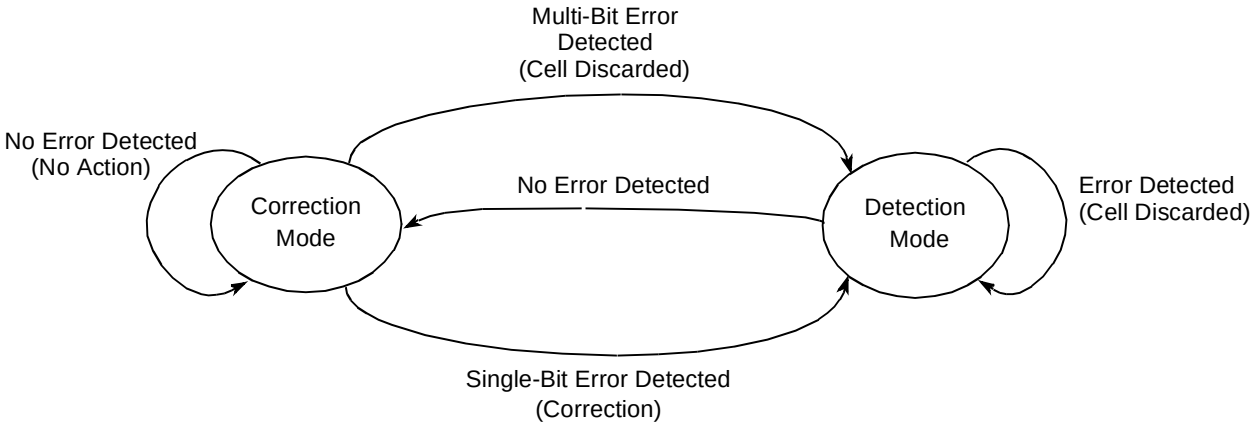


Figure 33-6. HEC: Receiver Modes of Operation

The MTC can also perform idle/unassigned cell filtering. Both features are programmable (MTC_MODE[CF]). Cells that are detected to be idle/unassigned are discarded, that is, not forwarded to the ATM controller in the QUICC Engine module. The MTC receive function also includes cell statistics and maintains counters for received cells (RCC), corrected cells (CCC), filtered cells (FCC) and erroneous cells (ECC). The counters are active when the cell delineation is in the SYNC state, each counter is 16 bits wide and when a counter overflows an interrupt can optionally be generated to the core.

33.3 SAM Programming Model

The MTC parameters are stored in the multi-user RAM (MURAM), Figure 33-7 shows the relationship between internal and external parameters controlled by an MTC:

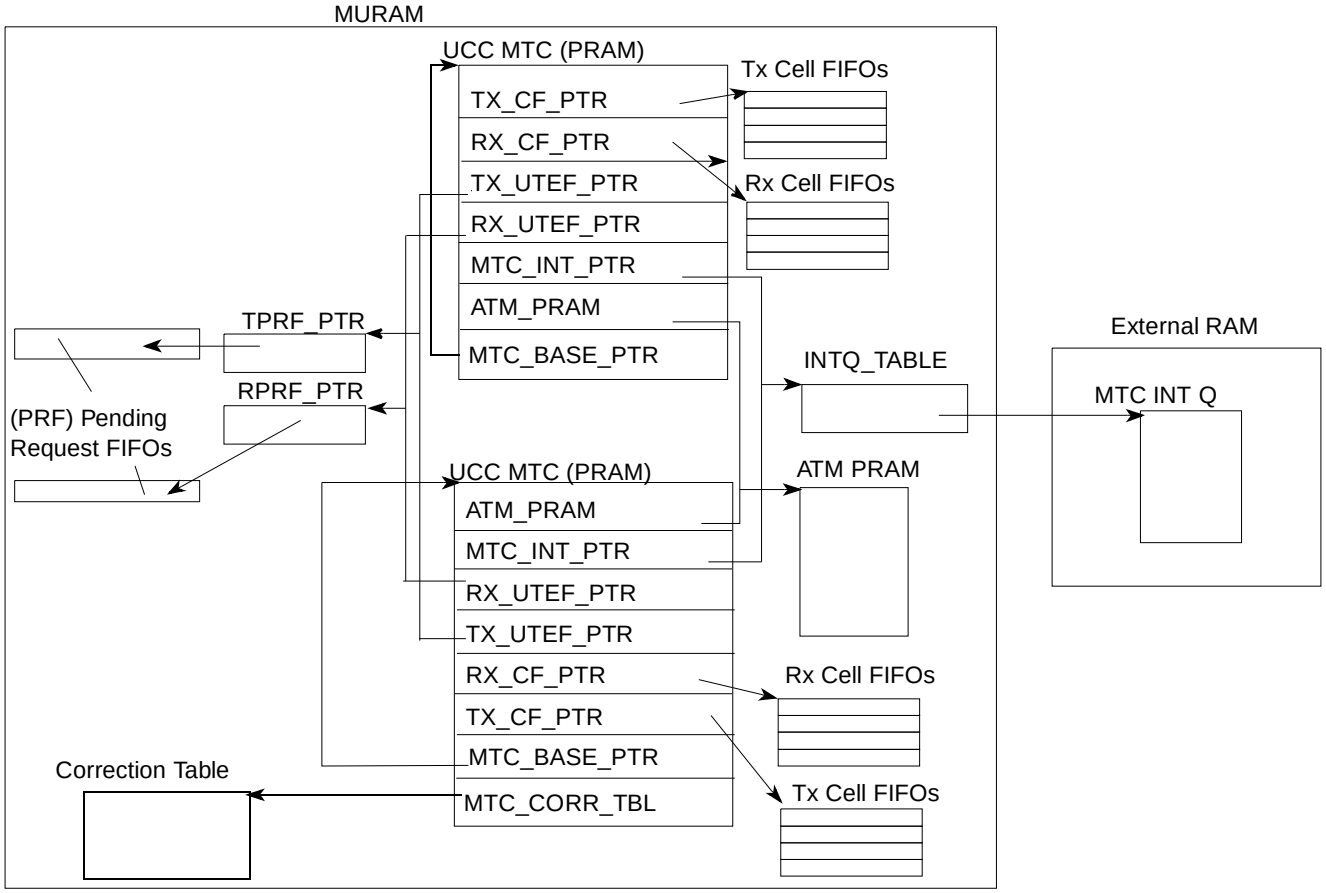


Figure 33-7. MTC Parameters

Each MTC PRAM requires 256 bytes of MURAM, in addition the MTC PRAM must be 256 bytes aligned. The MTC_UCC_BASE_PTR points to the base of the MTC PRAM for this UCC-controlled MTC. It is recommended but not mandatory that this pointer point to the base of the UCC PRAM for the UCC in serial ATM mode.

Figure 33-7 shows all parameters that consume MURAM bytes with the MTC PRAM. These include cell FIFOs, correction table, the UTEF and its associated pending request FIFO (PRF), the interrupt queue table, and the ATM PRAM. Each MTC PRAM controls both a transmitter and a receiver for one TC layer. Each MTC has both an Rx and a Tx cell FIFO ring of programmable depth. These cell FIFOs cannot be shared with another MTC. There is a pointer in the MTC PRAM to a transmit UTEF and a receive UTEF can be shared with other MTCs transmitters and receivers, respectively. Configure the TX_UTEF_PTR or RX_UTEF_PTR to reference the same UTEF. The transmit UTEF and Rx UTEF are separate structures and cannot be shared between Tx and Rx. Each UTEF table contains pointers to either an RxPRF or a TxPRF.

Each MTC PRAM has a pointer, MTC_INT_PTR, that points to the interrupt queue table pointers. Multiple MTCs can use the same interrupt queue in external memory. Each MTC can have its own interrupt queue or its interrupts can be directed to a shared queue, which allows the application to prioritize interrupts accordingly. Note that the interrupt queue for an MTC must reside in external memory.

With respect to the ATM controller, each MTC is essentially a PHY, and therefore an MTC has an associated ATM PRAM that can be shared between MTCs or unique to an MTC. There is a direct relationship between the ATM PRAM and an UTEF. MTCs cannot use a common UTEF to reference different ATM PRAM peripherals. If MTCs are configured to use the same UTEF, they must reference the same ATM PRAM.

If correction is enabled (MTC_MODE[SBC] = 0), a pointer, MTC_CORR_TBL, is required for the correction table. All MTCs that require correction should configure MTC_CORR_TBL to the same value.

33.3.1 MTC PRAM

Table 33-1 shows the MTC PRAM.

Table 33-1. MTC Parameter RAM Map

Offset[1]	Name	Width	Description
0x00	MTC_MODE	Hword	MTC Mode. See Section 33.3.1.1, “MTC Mode Register MTC_MODE.”
0x02—0x05	—	—	Reserved. Initialize to 0
0x06	MTC_RCC	Hword	MTC Received Cells Counter. Counts the number of received cells that are neither filtered, corrected or have any errors. This counter is only active when the cell delineation is in the SYNC state.
0x08	MTC_INT_MASK	Hword	MTC Interrupt Mask. This interrupt mask has the same bit definitions as an MTC Interrupt Queue Entry Event field. See Section 33.3.1.4, “MTC Interrupt Table And Queues,” for more details.
0x0A	MTC_FCC	Hword	MTC Filtered Cells Counter. Counts the number of filtered cells that are received by this TC layer. If MTC_MODE[CF] allows the cell to pass through without filtering then the cell is counted in MTC_RCC. This counter is only active when the cell delineation is in the SYNC state.
0x0C	MTC_CCC	Hword	MTC Corrected Cells Counter. Counts the number of corrected cells received by this TC layer. If MTC_MODE[SBC] = 1 then no cells are corrected and the MTC_ECC is used to count these cells. This counter is only active when the cell delineation is in the SYNC state and the ITU-T I.432 specification correction mode.
0x0E	MTC_ICC	Hword	MTC Idle Cells Counter. Counts the number of idle cells generated by the TC layer transmitter when the Tx Cell FIFO ring is empty. Idle cells generated by the ATM layer are counted here as well as those generated during underrun condition.
0x10	MTC_TCC	Hword	MTC Transmitted Cells Counter. Counts the number of data cells transmitted by this TC layer.
0x12	MTC_ECC	Hword	MTC Errored Cells Counter. Counts the number of cells with multiple bit HEC errors or single bit errors that cannot be corrected. This counter is only active when the cell delineation is in the SYNC state.

Table 33-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x14	MTC_STATE_TX	Hword	MTC State Transmit. Initialize to 0x8800. See Section 33.3.1.2, "MTC_STATE_TX Register."
0x16	MTC_STATE_RX	Hword	MTC State Receive. Initialize to 0x0000. See Section 33.3.1.3, "MTC_STATE_RX Register."
0x18-0x1B	—	—	Reserved. Initialize to 0.
0x1C	MTC_UCC_BASE_PTR	Word	MTC UCC Parameters Base Pointer. Offset in the MURAM where the MTC PRAM begins. This parameter must point to a 256 byte aligned address and is valid only when the UCC controls an MTC. This parameter is offset from the UCC parameter base. The recommended value is the PRAM base for the UCC controlling the MTC.
0x20	MTC_INT_PTR	Word	MTC Layer Interrupt Table Pointer. Pointer to a MURAM offset where the MTC Interrupt Tables reside. It is recommended that all MTCs program MTC_INT_PTR to the same value. This parameter must be 16 byte aligned. See Section 33.3.1.4, "MTC Interrupt Table And Queues."
0x24	MTC_SCTL	Byte	MTC SDMA Control. Controls the SDMA options for interrupts written to the interrupt queue. See Section 33.3.1.6, "MTC Interrupt Queue SDMA Control."
0x25	MTC_INT_NUM	Byte	MTC Interrupt Number. Required for identifying entries written to the interrupt queue, identifies the source MTC of an interrupt entry in the interrupt queue. See Section 33.3.1.4, "MTC Interrupt Table And Queues."
0x26	MTC_INT_TOF	Byte	MTC Interrupt Transmit Table Offset. Defines which Interrupt Table this MTC uses for the transmitter. The Interrupt Queue Table use by this MTC is found by MTC_INT_PTR + MTC_INT_TOF * 16. It is therefore possible for each MTC to have a separate interrupt queue by programming MTC_INT_TOF to differing values, or to share an interrupt queue setting MTC_INT_TOF to the same value in the MTCs' PRAM. The MTC_INT_TOF must be use a different table than that used by MTC_INT_ROF. It is not possible for any MTC receiver to access the interrupt queue used by any MTC transmitter. See Section 33.3.1.4, "MTC Interrupt Table And Queues."
0x27	MTC_INT_ROF	Byte	MTC Interrupt Receive Table Offset. Defines which Interrupt Table this MTC uses for the receiver. The Interrupt Queue Table use by this MTC is found by MTC_INT_PTR + MTC_INT_ROF * 16. It is therefore possible for each MTC to have a separate interrupt queue by programming MTC_INT_ROF to differing values, or to share an interrupt queue setting MTC_INT_ROF to the same value in the MTCs' PRAM. The MTC_INT_ROF must be use a different table than that used by MTC_INT_TOF. No MTC transmitter can access the interrupt queue used by any MTC receiver. See Section 33.3.1.4, "MTC Interrupt Table And Queues."
0x28-0x3F	—	—	Reserved. Initialize to 0
0x40	MTC_TX_SCRAM_HI	Word	MTC Transmit Scrambler State High Word. Microcode managed parameter initialize to 0.
0x44	MTC_TX_SCRAM_LO	Word	MTC Transmit Scrambler State Low Word. Microcode managed parameter initialize to 0.

Table 33-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x48	MTC_TX_ATM_PRAM	Hword	Transmitter ATM PRAM. Identifies which available thread is used to serve as the ATM Parameter RAM. MTC_RX_ATM_PRAM should be programmed to access the same PRAM area as MTC_TX_ATM_PRAM. See Section 33.3.1.7, "MTC Transmit ATM PRAM—MTC_TX_ATM_PRAM."
0x4A	MTC_TX_COSET	Byte	MTC Transmit Coset. XORed with cell HEC before transmission when MTC_MODE[TC] = 0. The recommended value is 0x55. The least significant bit of MTC_TX_COSET is XORed with the least significant bit of the cell HEC. Similarly, the most significant bit is XORed with the HEC msb.
0x4B-0x4F	—	—	Reserved. Initialize to 0
0x50	MTC_TX_CF_BP	Hword	MTC Transmit Cell FIFO Ring Base Pointer. Pointer to MURAM address that is the start of the address of this cell FIFO ring. Must point to a 64 byte aligned address. See Section 33.3.1.8, "Transmit Cell FIFO Ring."
0x52	MTC_TX_CF_EP	Hword	MTC Transmit Cell FIFO Ring End Pointer. Pointer to MURAM address that is the end address of this MTC cell FIFO ring. No entry exists at this address. Must point to a 64 byte-aligned address within the MURAM.
0x54	MTC_TX_CF_FP	Hword	MTC Transmit Cell FIFO Ring Fill Pointer. Updated by the MTC when the ATM controller delivers a cell to the FIFO ring. Microcode managed parameter; initialize to MTC_TX_CF_BP.
0x56	MTC_TX_CF_XP	Hword	MTC Transmit Cell FIFO Ring Extract Pointer. Updated by the MTC whenever a cell is fully transmitted via the SI. Microcode managed parameter; initialize to MTC_TX_CF_BP.
0x58	MTC_TX_PAGE	Word	MTC Transmit Page. Must be programmed to the page base of the ATM controller that this MTC accesses. For example, if this MTC uses UCC2 Tx as the ATM controller SNUM and the default page has not been changed for UCC2 through the Assign Page Host Command, program MTC_TX_PAGE to 0x00008500.
0x5C-0x63	—	—	Reserved. Initialize to 0
0x64	MTC_TX_MPHY_ADD	Word	MTC Transmit Multi Phy Address. Required in order to assign an APC table to this MTC. See Section 33.3.1.9, "MTC Transmit Multi PHY Address—MTC_TX_MPHY_ADD."
0x68	MTC_TX_PLD_PTR	Word	MTC Transmit Payload Pointer. Microcode managed parameter; initialize to 0.
0x6C-0x6E	—	—	Reserved. Initialize to 0
0x6F	MTC_TX_PLD_CTR	Byte	MTC Transmit Payload Counter. Microcode managed parameter; initialize to 0.
0x70	MTC_TX_UTEF_PTR	Word	MTC Transmit Utopia Emulation Pointer. Pointer to a MURAM address that contains the Utopia emulation FIFO table for this MTC. This parameter must be programmed to a 16-byte aligned address. See Section 33.3.1.10, "MTC Transmit Utopia Emulation FIFO."
0x74	MTC_TX_HEC	Byte	MTC Transmit HEC. Microcode managed parameter initialize to 0.
0x75-0x9F	—	—	Reserved. Initialize to 0.

Table 33-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0xA0	MTC_RX_SCRAM_HI	Word	MTC Receive Scrambler State High Word. Microcode managed parameter; initialize to 0.
0xA4	MTC_RX_SCRAM_LO	Word	MTC Receive Scrambler State Low Word. Microcode managed parameter; initialize to 0.
0xA8	MTC_RX_ATM_PRAM	Hword	Receiver ATM PRAM. Identifies which available thread is used to serve as the ATM Parameter RAM. It is recommended that MTC_TX_ATM_PRAM is programmed to access the same PRAM area as MTC_RX_ATM_PRAM. See Section 33.3.1.11, “MTC Receive ATM PRAM—MTC_RX_ATM_PRAM.”
0xAA	MTC_RX_COSET	Byte	MTC Receive Coset. XORed with cell HEC before HEC validation when MTC_MODE[RC] = 0. Recommended value is 0x55, the least significant bit of MTC_RX_COSET is XORed with the least significant bit of the Rx cell HEC, similarly the most significant bit is XORed with the HEC msb.
0xAB	MTC_RX_CTR	Byte	MTC Receive Counter. Microcode managed parameter initialize to 0.
0xAC	MTC_RX_PLD	Hword	MTC Receive Payload. Cell data temporary storage. Microcode managed parameter; initialize to 0.
0xAE	MTC_SHIFT_CTR_LFT	Byte	MTC Shift Counter Left. Required for cell delineation. Microcode managed parameter; initialize to 0.
0xAF	MTC_SHIFT_CTR_RGT	Byte	MTC Shift Counter Right. Required for cell delineation. Microcode managed parameter; initialize to 0.
0xB0	MTC_RX_CF_BP	Hword	MTC Receive Cell FIFO Ring Base Pointer. Pointer to MURAM address that is the start of the address of this cell FIFO ring. Must point to a 64-byte aligned address. See Section 33.3.1.12, “Receive Cell FIFO Ring,” for more details.
0xB2	MTC_RX_CF_EP	Hword	MTC Receive Cell FIFO Ring End Pointer. Pointer to MURAM address that is the end address of this MTC cell FIFO ring. No entry exists at this address. Must point to a 64 byte-aligned address within the MURAM.
0xB4	MTC_RX_CF_FP	Hword	MTC Receive Cell FIFO Ring Fill Pointer. Updated by the MTC when the MTC controller delivers a full cell to the FIFO ring. Microcode managed parameter; initialize to MTC_RX_CF_BP.
0xB6	MTC_RX_CF_XP	Hword	MTC Receive Cell FIFO Ring Extract Pointer. Updated by the ATM Controller whenever a cell is passed to the ATM layer. Microcode managed parameter; initialize to MTC_RX_CF_BP.
0xB8	MTC_RX_HEC	Byte	MTC Receive HEC. Microcode managed parameter; initialize to 0.
0xB9-0xBB	—	—	Reserved. Initialize to 0
0xBC-0xBF	MTC_RX_PAGE	Word	MTC Receive Page. Must be programmed to the page base of the ATM controller that this MTC accesses. For example, if this MTC uses UCC3 Rx as the ATM controller SNUM and the default page has not been changed for UCC3 through the Assign Page Host Command, program MTC_RX_PAGE to 0x00008600.
0xC0	MTC_RX_MPHY_ADD	Word	MTC Receive Multi Phy Address. Required to perform address compression correctly for this MTC. See Section 33.3.1.13, “MTC Receive Multi PHY Address—MTC_RX_MPHY_ADD and MTC_RX_MPHY_ADD_EXT.”

Table 33-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0xC4	MTC_RX_MPHY_ADD_EXT	Word	MTC Receive Multi Phy Address Extension. Required in order to correctly perform address compression for this MTC. See Section 33.3.1.13, "MTC Receive Multi PHY Address—MTC_RX_MPHY_ADD and MTC_RX_MPHY_ADD_EXT."
0xC8	MTC_RX_PLD_PTR	Word	MTC Receive Payload Pointer. Microcode managed parameter; initialize to 0.
0xCC-0xCE	—	—	Reserved. Initialize to 0.
0xCF	MTC_RX_PLD_CTR	Byte	MTC Receive Payload Counter. Microcode managed parameter; initialize to 0.
0xD0	MTC_RX_UTEF_PTR	Word	MTC Receive Utopia Emulation Pointer. Pointer to a MURAM address that contains the Utopia Emulation FIFO Table for this MTC. This parameter must be programmed to a 16 byte-aligned address. See Section 33.3.1.14, "MTC Receive Utopia Emulation FIFO."
0xD4	MTC_RX_CD_PLD	Word	MTC Receiver Cell Delineation Payload. Remainder after cell delineation. Microcode managed parameter initialize to 0.
0xD8	MTC_CD_CTR	Byte	MTC Cell Delineation Counter. Microcode managed parameter; initialize to 0.
0xD9	MTC_ALPHA	Byte	MTC Alpha. ITU-T I.432 Alpha; recommended value is 7.
0xDA	MTC_ALPHA_CTR	Byte	MTC Alpha Counter. Up counter for ALPHA; initialize to 0.
0xDB	MTC_DELTA	Byte	MTC Delta. ITU-T I.432 Delta; recommended value is 6.
0xDC	MTC_DELTA_CTR	Byte	MTC Delta Counter. Up counter for DELTA; initialize to 0.
0xDD-0xDF	—	—	Reserved. Initialize to 0.
0xE0	MTC_CORR_TBL	Word	MTC Correction Table. Pointer to the correction table that is valid only when MTC_MODE[SBC] = 1. Must point to a 256 byte-aligned address. See Section 33.3.1.15, "MTC Correction Table—MTC_CORR_TBL."
0xE4-0xFF	—	—	Reserved. Initialize to 0.

Note:

1 Offset from UCCx PRAM when using UCC and MTC_BASE_PTR = UCCx PRAM Base.

33.3.1.1 MTC Mode Register MTC_MODE

Each MTC layer block is configured using a TC layer mode register MTC_MODE, as shown in Figure 33-8.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RXEN	TXEN	RPS	TPS	RC	TC	SBC	CF		BBS	—			IMA	RAD	THD
Offset	0x00															

Figure 33-8. MTC Mode Register MTC_MODE

Table 33-2 describes the MTC_MODE fields.

Table 33-2. MTC_MODE Field Descriptions

Bits	Name	Description
0	RXEN	MTC Layer Rx enable bit. Enables the MTC receiver: 0 MTC Layer Rx operation is disabled. 1 MTC Layer Rx operation is enabled.
1	TXEN	MTC Tx enable bit. Enables the MTC transmitter: 0 MTC Tx operation is disabled. 1 MTC Tx operation is enabled.
2	RPS	Rx Payload DeScrambling 0 Payload descrambling is performed on received payload data. 1 No payload descrambling is performed on received payload data.
3	TPS	Tx Payload Scrambling 0 Payload scrambling is performed on transmitted payload data. 1 No payload scrambling is performed on transmitted payload data.
4	RC	Rx Coset Enable 0 XOR with RX_COSET is done on received HEC. 1 No XOR with RX_COSET is done on received HEC.
5	TC	Tx Coset Enable 0 XOR with TX_COSET is done on transmitted HEC. 1 No XOR with TX_COSET is done on transmitted HEC.
6	SBC	Header Single-Bit error Correction 0 Perform single-bit error correction on the header according to HEC while in SYNC state. 1 Do not perform single-bit error correction on the header. When this bit is cleared, the correction table referenced by MTC_CORR_TBL must be initialized.
7–8	CF	Rx Idle/Unassigned Cells Filtering 00 No cell filtering on Rx cells. 01 Idle cell filtering—idle cells are discarded. 10 Unassigned cell filtering—unassigned cells are discarded. 11 Idle and unassigned cell filtering—both idle and unassigned cells are discarded. The Header of idle cell (ITU-T I.361): b00000000_00000000_00000000_00000001 The Header of unassigned cell (ITU-T I.361): b00000000_00000000_00000000_0000xxx0 Note that physical layer cells bypass the TC layer; they are not filtered. Also note that the filter works on the header only and ignores the HEC.
9	BBS	Bit/Byte Synchronous mode for cell delineation. 0 Bit synchronous. The ATM cells starts at an arbitrary offset within the TDM frame. 1 Byte synchronous. The ATM cell is byte aligned within the TDM frame.
10–12	—	Reserved. Initialize to 0.
13	IMA	IMA mode 0 Rx is not in IMA. 1 Rx is in IMA mode.
14	RAD	Receive ATM Disable 0 Pass cells, except filtered cells, to the ATM or IMA microcode. 1 No cells are passed to the ATM or IMA microcode. The MTC will maintain the cell delineation algorithm and counters will be updated.

Table 33-2. MTC_MODE Field Descriptions (continued)

Bits	Name	Description
15	THD	Transmit HEC disable. 0 Transmit HEC 1 HEC transmission disabled.

33.3.1.2 MTC_STATE_TX Register

MTC_STATE_TX, shown in Figure 33-9, holds the current state of the MTC transmitter. It is required for transmit state fields that require coherency in a multi-RISC environment.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MTCI	THEC	TIERU	IDL_INT	IDLE_C	TSTRTD	IOF_INT	—								
Offset	0x14															

Figure 33-9. MTC_STATE_TX Register

Table 33-3 describes the MTC_STATE_TX fields.

Table 33-3. MTC_STATE_TX Field Descriptions

Bits	Name	Description
0	MTCI	MTC Transmit Idle. Reserved for internal use. Initialize to 1.
1	THEC	Transmit HEC. Reserved for internal use. Initialize to 0.
2	TIERU	Transmit Interrupt Event Register Update. Reserved for internal use. Initialize to 0.
3	IDL_INT	Idle Interrupt. Reserved for internal use. Initialize to 0.
4	IDLE_C	Idle Check. Reserved for internal use. Initialize to 1.
5	TSTRTD	Transmit Started. Reserved for internal use. Initialize to 0.
6	IOF_INT	IOF Interrupt generation. Reserved for internal use. Initialize to 0.
7–15	—	Reserved. Initialize to 0.

33.3.1.3 MTC_STATE_RX Register

MTC_STATE_RX, shown in Figure 33-10, holds the current state of the MTC receiver. It is used to hold receive state fields that require coherency in a multi-RISC environment.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	CFOV	RHEC	WAIT	CM	CDS		RIERU	CFOV_C	—			RCC				—
Offset	0x16															

Figure 33-10. MTC_STATE_RX Register

Table 33-4 describes the MTC_STATE_RX fields.

Table 33-4. MTC_STATE_RX Field Descriptions

Bits	Name	Description
0	CFOV	Cell FIFO Overflow. Reserved for internal use. Initialize to 0.
1	RHEC	Receive HEC. Reserved for internal use. Initialize to 0.
2	WAIT	Cell Delineation Wait. Reserved for internal use. Initialize to 0.
3	CM	Correction Mode. 0 Detection mode. 1 Correction mode. Reserved for internal use. Initialize to 0.
4–5	CDS	Cell Delineation State. 00 HUNT state. 01 PRESYNC state. 10 Reserved. 11 SYNC state. Reserved for internal use. Initialize to 0.
6	RIERU	Receive Interrupt Event Register Update. Reserved for internal use. Initialize to 0.
7	CFOV_C	Cell FIFO Overflow Check. Reserved for internal use. Initialize to 0.
8–10	—	Reserved. Initialize to 0.
11–14	RCC	Receive Counter Code. 0000 Reserved. 0001 Reserved. 0010 Reserved. 0011 MTC_RCC. 0100 Reserved. 0101 MTC_FCC. 0110 MTC_CCC. 0111 Reserved. 1000 Reserved. 1001 MTC_ECC. 1010 Reserved. 1011 Reserved. 1100 Reserved. 1101 Reserved. 1110 Reserved. 1111 Reserved. Reserved for internal use. Initialize to 0.
15	—	Reserved. Initialize to 0.

33.3.1.4 MTC Interrupt Table And Queues

Each MTC interrupt queue holds the interrupt entries for one or more MTCs. Each interrupt queue is controlled through an interrupt table in the MURAM. The interrupt tables must be 16 bytes aligned. Table 33-5 defines the MTC interrupt table parameters. For the application to prioritize interrupts, the SAM supports up to 256 MTC interrupt tables controlled by the MTC_INT_TOF and MTC_INT_ROF in the MTC PRAM. A transmit and receive MTC cannot reference the same MTC interrupt table. Also, two

transmit (or receive) MTCs on differing QUICC Engine peripherals cannot access the same MTC interrupt table.

Table 33-5. MTC Interrupt Table

Offset[1]	Name	Width	Description
0x00	MTC_INTQ_BASE	Word	MTC or MTCs Interrupt Queue Base. Pointer to external memory to the interrupt queue base for this MTC. This address must be 4 bytes aligned.
0x04	MTC_INTQ_PTR	Word	MTC Interrupt Queue Pointer. Current interrupt queue entry offset. Microcode managed parameter initialize to MTC_INTQ_BASE.
0x08	MTC_INTQ_ENT	Word	MTC Interrupt Queue Entry. Holds the interrupt queue entry. Microcode managed parameter, initialize to 0.
0x0C	MTC_INTE_CTL	Byte	MTC Interrupt Event Control. Used to identify which interrupt bit in the UCCE this MTC Interrupt Table Uses. See Section 33.3.1.5, “Event Registers” for more details.
0x0D-0x0F	—	—	Reserved. Initialize to 0.

Note:

1 Offset from MTC_INT_PTR + MTC_INT_TOF * 16. Must be 16 bytes aligned in the MURAM.

Figure 33-11 shows the format of an interrupt entry written to the MTC Interrupt Queue.

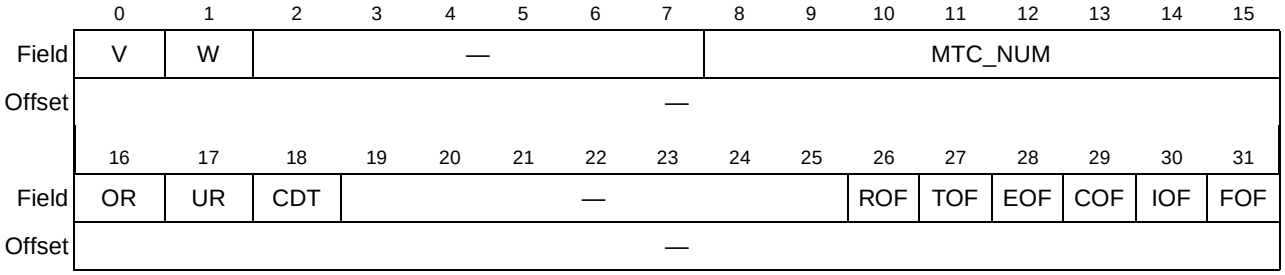


Figure 33-11. MTC Interrupt Queue Entry

Table 33-6 shows the MTC interrupt queue entry field definitions. More than one interrupt bit can be set in the same entry in the interrupt queue.

Table 33-6. MTC Interrupt Queue Field Descriptions

Bits	Name	Description
0	V	Valid. 0 This interrupt entry is not valid; no interrupt entry is present. The host should initialize each entry’s V bit to 0. 1 This interrupt entry is valid and the host can process this interrupt entry. The host should clear this field after reading this interrupt entry.
1	W	Wrap. 0 This is not the last entry in the interrupt queue. 1 This is the last entry in the interrupt queue, wraps back MTC_INTQ_BASE after an interrupt is written to the entry with W = 1. The host should initialize this field.
2–7	—	Reserved. Initialize to 0.

Table 33-6. MTC Interrupt Queue Field Descriptions (continued)

Bits	Name	Description
8–15	MTC_NUM	MTC Number. This field is copied to the interrupt entry from the MTC_INT_NUM parameter of the MTC PRAM. This field is used to identify which MTC this interrupt entry originates from.
16	OR	Overrun. Rx FIFO OverFlow. Set when Rx FIFO is full and another cell is received. The cell is discarded.
17	UR	Underrun. No ATM cell to transmit. Set when the Tx FIFO is empty and the transmission of a cell is completed. An idle cell is sent. The idle cell header is: 0x00000001 (I.432), whose HEC is: 0x52. The idle cell payload is:0x6A (I.432).
18	CDT	Cell delineation toggled. Set when the cell delineation transitions to or from the SYNC state.
19–25	—	Reserved. Initialize to 0.
26	ROF	Received cell counter overflow Set when the received cells counter passes its maximum value.
27	TOF	Transmitted cell counter overflow Set when the transmitted cells counter passes its maximum value.
28	EOF	Errored cells counter overflow Set when the errored cells counter passes its maximum value.
29	COF	Corrected cells counter overflow. Set when the corrected cells counter passes its maximum value.
30	IOF	Tx Idle cells counter overflow. Set when the Tx idle cells counter passes its maximum value.
31	FOF	Filtered cells counter overflow. Set when the filtered cells counter passes its maximum value.

33.3.1.5 Event Registers

Because the UCC must be configured as a slow communications controller for SAM, its event register is 16 bits. The corresponding UCCM bit must be set if the interrupt is to be issued to the core. Figure 33-12 shows the UCC event and mask register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	GUN	GOV
MTC_INTE_CTL	0x0	0x0	0x1	0x1	0x2	0x2	0x3	0x3	0x4	0x4	0x5	0x5	0x6	0x6	—	—

Figure 33-12. UCC Event/Mask Register

Table 33-7 describes the MTC event register interrupt bits.

Table 33-7. MTC Event Register For UCC

Bits	Name	Description
0, 2, 4, 6, 8, 10, 12	QOV	Queue Overflow. When this bit is set the interrupt queue corresponding to the MTC_INTE_CTL setting has experienced an overflow.

Table 33-7. MTC Event Register For UCC (continued)

Bits	Name	Description
1, 3, 5, 7, 9, 11, 13	MTCE	MTC Event. The interrupt queue corresponding to the MTC_INTE_CTL has experienced an interrupt event.
14	GUN	Global Underrun. The UCC Transmit FIFO has experienced an underrun. The UCC should be disabled via the SIMxR and reinitialized.
15	GOV	Global Overrun. The UCC Receive FIFO has experienced an overrun. The UCC should be disabled via the SIMxR and reinitialized.

NOTE

If there is a QOV interrupt, an interrupt entry in the interrupt queue has been lost. This interrupt may be an OR event. This event effectively disables the MTC receiver. Therefore, after a QOV interrupt the MTC_STATE[CFOV] should be checked to determine whether the MTC receiver should be reinitialized and to determine whether the lost interrupt was a receive cell FIFO overrun.

33.3.1.6 MTC Interrupt Queue SDMA Control

Figure 33-13 shows configuration information for the MTC_SCTL.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		—		BIB
Offset	0x24							

Figure 33-13. MTC_SCTL—MTC Interrupt Queue SDMA Control Register

Table 33-8 describes the MTC_SCTL bit fields.

Table 33-8. MTC_SCTL Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global. Asserting GBL enables snooping of the MTC Interrupt Queue. To use Global mode, SDMR[GBL_1_MSK] must also be set.
3–4	BO	Byte ordering. 00, 01, 11 Reserved 10 Big endian
5–6	—	Reserved, should be cleared.
7	BIB	MTC Interrupt Queue Bus: 0 Resides on the CBS bus. 1 Resides on the secondary bus.

33.3.1.7 MTC Transmit ATM PRAM—MTC_TX_ATM_PRAM

Each MTC has an associated ATM controller. The ATM controller PRAM area used by the MTC can be controlled by any unused UCC transmit SNUM. Figure 33-14 shows the ATM_TX_ATM_PRAM parameter.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	UT_SNUM															
Offset	0x48															

Figure 33-14. MTC_TX_ATM_PRAM

Table 33-9 describes the fields of the MTC_TX_ATM_PRAM.

Table 33-9. MTC_TX_ATM_PRAM Field Descriptions

Bits	Name	Description
0–15	UT_SNUM[1]	Unused UCC Transmit SNUM. Used to identify the SNUM of the UCC that serves as the ATM PRAM for this MTC 0x0000 UCC1, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0001 0x0010 UCC2, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0011 0x0020 UCC3, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0021 0x0030 UCC4, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0031 0x0040 UCC5, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0041

Note:

1 The UCC SNUM used must not be that of an enabled UCC. The GUMR_L[ENT, ENR] must both be programmed to 0 for any UCC used as the ATM PRAM for SAM based TC layers.

33.3.1.8 Transmit Cell FIFO Ring

A transmit cell FIFO ring is unique to one MTC. The transmit cell FIFO ring configuration is shown in Figure 33-15.

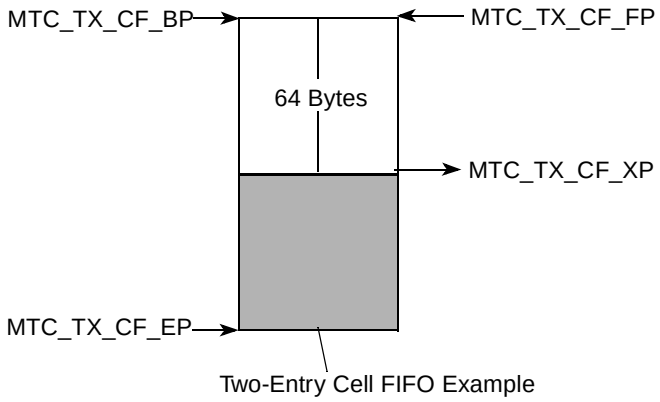


Figure 33-15. MTC Transmit Cell FIFO Ring

Each FIFO entry is 64 bytes. The example shows the configuration required to initialize a two-cell FIFO. The MTC_TX_CF_EP points to the end of the table; it does not point to the beginning of the last entry.

The MTC_TX_CF_BP must point to a 64-byte aligned address, as is also true for the MTC_TX_CF_EP. The transmit cell FIFO ring should be two entries long.

33.3.1.9 MTC Transmit Multi PHY Address—MTC_TX_MPHY_ADD

MTC_TX_MPHY_ADD is required in order to assign an APC table to this MTC. The MPHY address programmed in MTC_TX_MPHY_ADD must be unique to the ATM PRAM used by this MTC. Thus, if more than one MTC uses the same ATM PRAM, they must have different multi-PHY addresses. If these MTCs use their own or different ATM PRAM, they can use the same PHY address. An MTC using its own unique ATM PRAM is analogous to a single PHY configuration on the UTOPIA bus. Furthermore, a group of MTCs sharing the same ATM PRAM is similar to a multi-PHY configuration on the UTOPIA bus. Each ATM PRAM used has an associated PHY or group of PHYs, in essence TC layers, that it views as UTOPIA PHYs—even though no UTOPIA bus exists. Figure 33-16 shows the fields of the MTC_TX_MPHY_ADD.

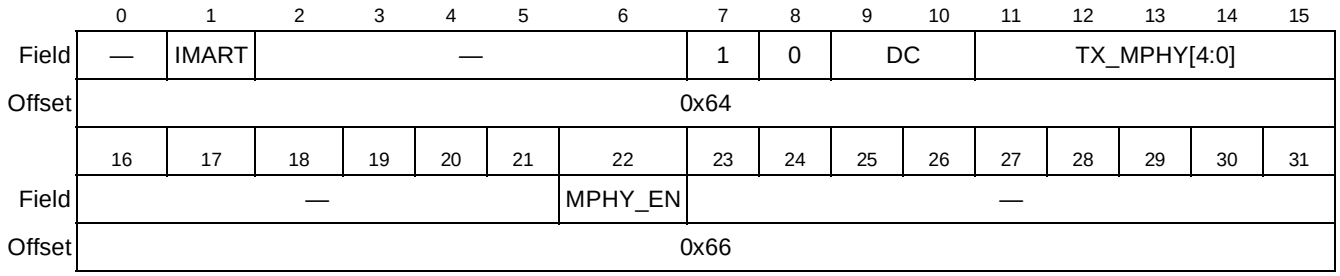


Figure 33-16. MTC_TX_MPHY_ADD Register

Table 33-10 describes the MTC_TX_MPHY_ADD fields.

Table 33-10. MTC_TX_MPHY_ADD Field Descriptions

Bits	Name	Description
0	—	Reserved, initialize to 0.
1	IMART	IMA Return 0 If transmitter operates in IMA mode, clear IMART. 1 If transmitter does not operate in IMA mode, set IMART.
2–6	—	Reserved, initialize to 0.
7	—	Must be initialized to 1.
8	—	Must be initialized to 0.
9–10	DC	Device Code. The two most significant bits of the TX_MPH. These bits identify the device on the UTOPIA bus to which this MTCs PHY number belongs. Required in order to identify the APC table for this MTC (PHY).
11–15	TX_MPHY[4:0]	Transmit Multi-PHY Address. Initialize to the PHY address for this MTC.
16-21	—	Reserved, initialize to 0.

Table 33-10. MTC_TX_MPHY_ADD Field Descriptions (continued)

Bits	Name	Description
22	MPHY_EN	Multi-PHY Enable. 0 Single-PHY mode. 1 Multi-PHY mode. This bit must be set when MTC_MODE[IMA] = 1. Use single PHY mode when the ATM PRAM associated with this MTC has no other connected MTCs.
23–31	—	Reserved, initialize to 0.

33.3.1.10 MTC Transmit Utopia Emulation FIFO

Each MTC transmit UTOPIA emulation FIFO is composed of two distinct parts, a UTEF table and a pending request FIFO (PRF). The UTEF table contains pointers that define the size of the PRF. The PRF holds requests from the MTCs that use this UTEF. Both the UTEF Table and the PRF are in the MURAM. More than one MTC can access the same UTEF.

Table 33-11. UTEF Table

Offset[1]	Name	Width	Description
0x00	MTC_TPRF_BP	Word	MTC Transmit Pending Request FIFO Base Pointer. Base address of the PRF. Each entry in the PRF is 4 bytes. This address must be 4 bytes aligned.
0x04	MTC_TPRF_EP	Word	MTC Transmit Pending Request FIFO End Pointer. End address of the PRF. The size of the PRF should be 1 greater than the number of associated MTCs Transmit Cell FIFO entries. For example, if t 2 MTCs use this UTEF and each of these MTCs Transmit Cell FIFO rings are 4 entries long, then the size of the table should be 9 entries, 36 bytes. This address must be 4 byte aligned and point to the end of the last entry in the PRF.
0x08	MTC_TPRF_FP	Word	MTC Transmit Pending Request FIFO Fill Pointer. Microcode managed parameter initialize to MTC_TPRF_BP.
0x0C	MTC_TPRF_XP	Word	MTC Transmit Pending Request FIFO Extract Pointer. Microcode managed parameter initialize to MTC_TPRF_BP.

Note:

1 Offset from MTC_TX_UTEF_PTR. Must be 16 bytes aligned in the MURAM.

The user must clear the PRF during initialization.

33.3.1.11 MTC Receive ATM PRAM—MTC_RX_ATM_PRAM

Each MTC has an associated ATM controller. The ATM controller PRAM area used by the MTC can be any unused UCC SNUM. Figure 33-17 shows the fields of the MTC_RX_ATM_PRAM.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	UR_SNUM															
Offset	0xa8															

Figure 33-17. MTC_RX_ATM_PRAM

Table 33-12 describes the MTC_RX_ATM_PRAM fields.

Table 33-12. MTC_RX_ATM_PRAM Field Descriptions

Bits	Name	Description
0–15	UR_SNUM[1]	Unused UCC Receive SNUM. Used to identify the SNUM of the UCC that serves as the ATM PRAM for this MTC 0x0001 UCC1, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0000 0x0011 UCC2, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0010 0x0021 UCC3, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0020 0x0031 UCC4, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0030 0x0041 UCC5, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0040

Note:

[1] The UCC SNUM used must not be that of an enabled UCC. The GUMR_L[ENT, ENR] must both be programmed to 0 for any UCC used as the ATM PRAM for SAM-based TC layers.

33.3.1.12 Receive Cell FIFO Ring

A receive cell FIFO ring is unique to an MTC. The receive cell FIFO ring configuration is shown in Figure 33-18.

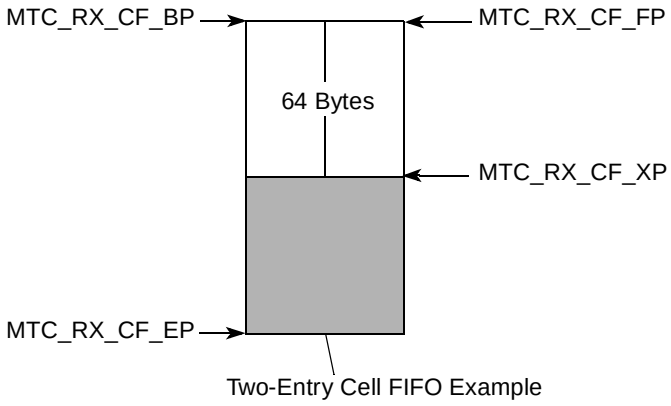


Figure 33-18. MTC Receive Cell FIFO Ring

Each FIFO entry is 64 bytes. The example shows the configuration required to initialize a two-cell FIFO. The MTC_RX_CF_EP points to the end of the table; it does not point to the beginning of the last entry. The MTC_RX_CF_BP must point to a 64-byte aligned address, as is also true for the MTC_RX_CF_EP. The receive cell FIFO ring should be two entries long.

33.3.1.13 MTC Receive Multi PHY Address—MTC_RX_MPHY_ADD and MTC_RX_MPHY_ADD_EXT

MTC_RX_MPHY_ADD, shown in Figure 33-19 is required to perform the address compression correctly for cells received by this MTC. The MPHY address programmed in MRX_RX_MPHY_ADD must be unique to the ATM PRAM used by this MTC. Thus, if more than one MTC uses the same ATM PRAM, they must have different multi-PHY addresses. If these MTCs use their own or different ATM PRAM, the same PHY address can be used for these MTCs.

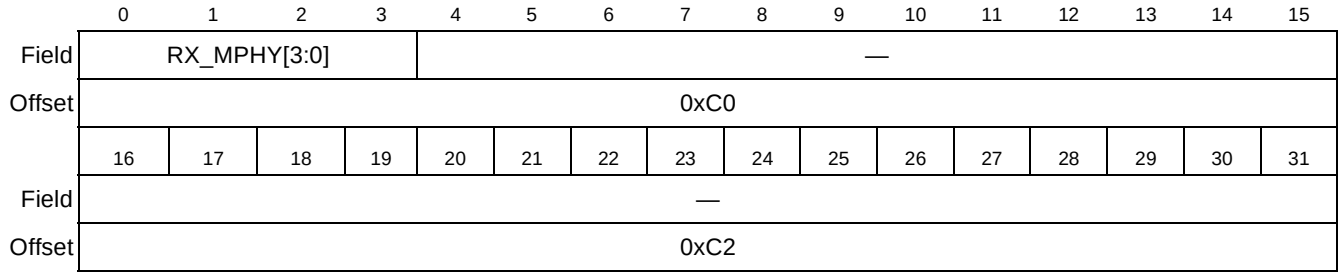


Figure 33-19. MTC_RX_MPHY_ADD Register

Table 33-13 describes the fields of the MTC_RX_MPHY_ADD register.

Table 33-13. MTC_RX_MPHY_ADD Field Descriptions

Bits	Name	Description
0–3	RX_MPHY[3:0]	Receive Multi PHY Address 3 to 0. Contains the least significant 4 bits of the UTOPIA emulation PHY address for this MTC. The most significant bit of the address is found in MTC_RX_MPHY_ADD_EXT[RX_MPHY[4]].
4–31	—	Reserved, initialize to 0.

Figure 33-20 shows the MTC_RX_MPHY_ADD_EXT.

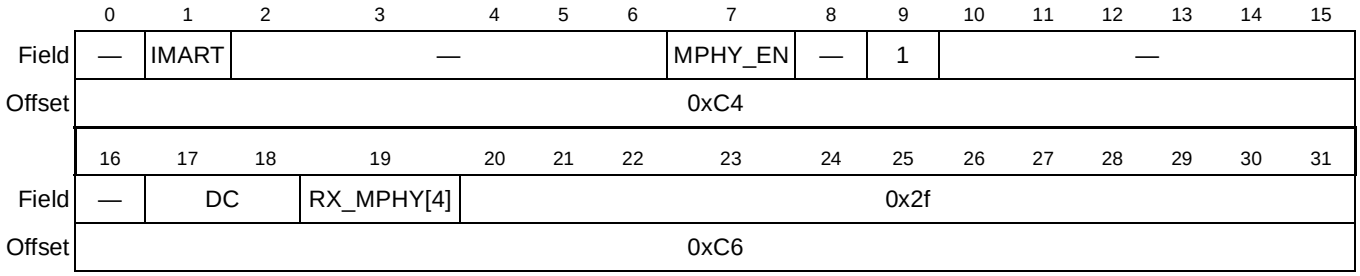


Figure 33-20. MTC_RX_MPHY_ADD_EXT Register

Table 33-14 describes the MTC_RX_MPHY_ADD_EXT fields.

Table 33-14. MTC_RX_MPHY_ADD_EXT Field Descriptions

Bits	Name	Description
0	—	Reserved, initialize to 0.
1	IMART	IMA Return 0 If MTC_MODE[IMA] = 1 set IMART to 0. 1 If MTC_MODE[IMA] = 0 set IMART to 1.
2–6	—	Reserved, Initialize to 0.
7	MPHY_EN	Multi-PHY Enable. 0 Single PHY mode. 1 Multi PHY mode. This bit must be set when MTC_MODE[IMA] = 1. Use single PHY mode when the ATM PRAM associated with this MTC has no other connected MTCs.
8	—	Reserved, initialize to 0.

Table 33-14. MTC_RX_MPHY_ADD_EXT Field Descriptions (continued)

Bits	Name	Description
9	—	Must be initialized to 1.
10–16	—	Reserved, initialize to 0.
17–18	DC	Device Code. Essentially the 2 most significant bits of the TX_MPHY, these bits identify which device on the UTOPIA bus this MTCs PHY number belongs to. Required for address compression.
19	RX_MPHY[4]	Receive Multi PHY Address 4. Most significant bit of the of the UTOPIA emulation PHY address for this MTC.
20–31	—	Reserved, initialize to 0x2f.

33.3.1.14 MTC Receive Utopia Emulation FIFO

Each MTC receive UTOPIA emulation FIFO is composed of two distinct parts, a UTEF table and a pending request FIFO (PRF). The UTEF table contains pointers that define the size of the PRF. The PRF holds requests from the MTCs that use this UTEF. Both the UTEF table and the PRF are in the MURAM. More than one MTC can access the same UTEF. The user must clear the PRF during initialization.

Table 33-15. UTEF Table

Offset[1]	Name	Width	Description
0x00	MTC_RPRF_BP	Word	MTC Receive Pending Request FIFO Base Pointer. Base address of the PRF. Each entry in the PRF is 4 bytes. This address must be 4 bytes aligned.
0x04	MTC_RPRF_EP	Word	MTC Receive Pending Request FIFO End Pointer. End address of the PRF. The size of the PRF should be 1 greater than the number of associated MTCs Receive Cell FIFO entries. For example, if there are 2 MTCs that use this UTEF and each of these MTCs Receive Cell FIFO rings are 4 entries long then the size of the table should be 9 entries, 36 bytes. This address must be 4 bytes aligned and point to the end of the last entry in the PRF.
0x08	MTC_RPRF_FP	Word	MTC Receive Pending Request FIFO Fill Pointer. Microcode managed parameter initialize to MTC_RPRF_BP.
0x0C	MTC_RPRF_XP	Word	MTC Receive Pending Request FIFO Extract Pointer. Microcode managed parameter initialize to MTC_RPRF_BP.

Note:

1 Offset from MTC_RX_UTEF_PTR. Must be 16 bytes aligned in the MURAM.

33.3.1.15 MTC Correction Table—MTC_CORR_TBL

The MTC correction table occupies 256 bytes of MURAM. Each entry in the table is one byte. The correction table allows the MTC to determine whether a cell has a HEC error and whether the error can be corrected. Correction occurs only for single bit errors. The correction table should be initialized only when the MTC_MODE[SBC] = 0. Table 33-16 shows the configuration requirements for the correction table.

Table 33-16. MTC Correction Table

Offset[1]	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	255	0	1	255	2	255	255	8	3	255	255	31	255	255	9	255
0x10	4	255	255	255	255	16	32	255	255	255	255	255	10	255	255	255
0x20	5	255	255	255	255	255	255	255	255	255	17	255	33	255	255	255
0x30	255	39	255	255	255	255	255	255	11	255	255	255	255	255	255	255
0x40	6	255	255	29	255	255	255	255	255	255	255	255	255	255	255	255
0x50	255	27	255	255	18	255	255	20	34	255	255	22	255	255	255	255
0x60	255	255	255	255	255	255	255	36	255	255	255	24	255	255	255	255
0x70	12	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0x80	7	255	255	255	255	255	30	255	255	15	255	255	255	255	255	255
0x90	255	255	255	255	255	255	255	255	255	255	255	38	255	255	255	255
0xA0	255	255	28	255	255	255	255	255	19	255	255	26	255	255	21	255
0xB0	35	255	255	255	255	255	23	255	255	255	255	255	255	255	255	255
0xC0	255	255	255	255	255	255	255	14	255	255	255	255	255	255	37	255
0xD0	255	255	255	255	255	255	25	255	255	255	255	255	255	255	255	255
0xE0	13	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
0xF0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

1 Offset from MTC_CORR_TBL. Must be 256 bytes aligned in the MURAM.

The correction table must be initialized to the values shown so that the MTC can use it to determine the location of any single-bit errors in the cell header and correct accordingly.

33.4 IMA Root Table SAM Programming

When the SAM is in use, the MTC ATM PRAM must configure the IMA root table IMACNTL parameter. This configuration is mandatory regardless of the setting of the value of MTC_MODE[IMA]. If MTC_MODE[IMA] = 0, only the IMACNTL[SAME] must be configured. However, as the ATM PRAM

IMAROOT is initialized, the full IMA root table (128 bytes) must be assigned and is therefore unavailable in the MURAM for other QUICC Engine peripherals. Figure 33-21 shows the IMACNTL.

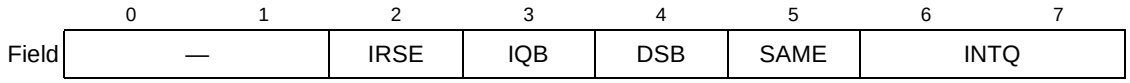


Figure 33-21. IMA Control (IMACNTL)

Table 33-17 describes the IMACNTL bit fields.

Table 33-17. IMACNTL Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	IRSE	IMA link receive statistics enable. If link receive statistics gathering is disabled, there is no need to initialize IRLINKSTAT or reserve space for the receive statistics table. 0 Link receive statistics gathering is disabled. 1 Link receive statistics gathering is enabled.
3	IQB	Interrupt queue bus. Defines on which bus the IMA interrupt queue is located. 0 On the CSB bus. 1 On the secondary bus.
4	DSB	Data structure bus. Defines on which bus the IMA external structure memory area is located. 0 On the CSB bus. 1 On the secondary bus.
5	SAME	Serial ATM Enable. 0 The TC layer is not SAM based 1 The TC layer is SAM based, such as a MTC.
6–7	INTQ	Number of the ATM interrupt queue dedicated to IMA events. Note that these do not include ICP cell reception events; the handling of ICP cell reception events is programmed in the RCT of the ICP channel defined by RICPCH.

33.5 ATM Controller UCC_Modes Initialization

Any SNUM of a UCC used by an MTC as the ATM controller must have the following bits in the UCC_Mode set accordingly:

- Non Pipelined UCC_Modes[NPL] = 1
- No multi-thread support UCC_Modes[Multi_Thread_En] = 0

Because the SAM requires any UCC SNUM used to serve as the ATM PRAM is not enabled (GUMR_L[ENT, ENR] = 0), there is no requirement to issue the ATM Transmit Command for this UCC. However, this command should be issued as part of the initialization to configure the UCC_Modes[UID] correctly. This command can be skipped only if the UCC_Modes[UID] is not required for ATM address compression.

33.6 UCC Initialization

The UCC must be configured for transparent protocol when it is connected to the SAM using the following initialization sequence:

1. Configure parallel I/O for the TDM interface required.
2. Initialize the SIU interrupt controller to mask or enable UCCE interrupts as appropriate.
3. Initialize the UCC registers. The UCC must be programmed to operate in QMC mode in the GUMR_L[MODE] and must be configured as a slow communications controller. Also, GUMR_L[16] should be set to enable serial ATM mode (see Table 23-3).
4. Initialize the SIRAM and SIRAM registers, except SIxGMR.
5. Initialize MTC PRAM.
6. Enable the TDM in the SIxGMR.
7. Enable the transmitter and receiver, GUMR_L[ENT, ENR] = 1.

33.6.1 Disabling the MTC

To disable the MTC transmitter, clear MTC_MODE[TXEN]. Similarly, to disable the receiver, clear MTC_MODE[RXEN]. After these bits are cleared, the MTC PRAM can safely be updated to its default state and the channel re-enabled only after the following conditions are true:

- If MTC TX is to be disabled, software must wait for MTC_STATE_TX[MTCI, IDLE_C] = 0 and then wait for MTC_TX_CF_FP = MTC_TX_CF_XP.
- If MTC_RX is to be disabled, software must wait for MTC_STATE_RX[CFOV_C] = 0 and then check that MTC_RX_CF_XP = MTC_RX_CF_FP.

To re-enable the MTC after the MTC PRAM is updated, set MTC_MODE[TXEN] and/or MTC_MODE[RXEN].

NOTE

It is possible to disable only the transmitter or the receiver and leave the other active. Clear MTC_MODE[TXEN] or MTC_MODE[RXEN] and update only the transmit or receive MTC parameters.

If the MTC and therefore an associated TDM is to be completely disabled, perform the following steps:

1. Clear MTC_MODE[TXEN] and MTC_MODE[RXEN].
2. Software must wait for MTC_STATE_TX[MTCI, IDLE_C] = 0 and then wait for MTC_TX_CF_FP = MTC_TX_CF_XP,
3. Software must wait for MTC_STATE_RX[CFOV_C] = 0 and then verify that MTC_RX_CF_XP = MTC_RX_CF_FP.
4. Disable TDM through the SIxGMR and clear both GUMR_L[ENT] and GUMR_L[ENR].

After these steps are completed, the MTC is entirely disabled and its PRAM area can be updated or reassigned as required. The UCC can then be assigned to another protocol.

33.6.2 Response to GUN or GOV

A GOV or GUN is a non-recoverable event for the UCC. After this event, the following procedure should be performed:

1. For GUN clear MTC_MODE[TXEN], for GOV clear MTC_MODE[RXEN].
2. For GUN disable the UCC transmitter by clearing GUMR_L[ENT], for GOV disable the UCC receiver by clearing GUMR_L[ENR]. If the entire MTC is to be stopped, the TDM should be disabled and both GUMR_L[ENT] and GUMR_L[ENR] cleared.
3. For GUN, software must wait for MTC_STATE_TX[MTCI, IDLE_C] = 0 and then wait for MTC_TX_CF_FP = MTC_TX_CF_XP. For GOV, software must wait for MTC_STATE_RX[CFOV_C] = 0 and then check that MTC_RX_CF_XP = MTC_RX_CF_FP
4. Reinitialize MTC PRAM for the affected UCC.
5. Re-enable any MTC by setting either GUMR_L[ENT] or GUMR_L[ENT].

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 single unified communication controller (UCC) and a time-division multiplexed (TDM) physical interface. Each UCC has its own parameter table and the channel's specific parameters. Each UCC also has its own event register, so the functionality of each UCC is orthogonal to that of adjacent UCCs. Therefore, theoretically it enables support of up to 512 serial channels provided sufficient QUICC Engine performance and that all UCCs can be dedicated for processing. The application must manage channel numbering since the numbering (0–63) is performed per UCC.

Each channel can be independently programmed to operate in either HDLC or transparent mode. Any available TDM in the serial interface (SI) can be used for the QMC protocol. The SI transfers data between the TDM interface and the UCC, which performs multiplexing/demultiplexing on the QMC channels. Figure 34-1 provides an overview of the QMC functionality.

Each UCC can work in QMC mode, either alone or together in any combination, spreading any of the 64 available QMC channels across the multiple UCCs. One TDM connection can be routed to one or more UCCs operating in QMC mode, with each UCC operating on different time slots.

Multiple TDMs can be used for QMC with combined routing to one UCC or separate UCCs. When multiple TDMs are connected to the same UCC, restrictions such as using common clocks and sync inputs apply; to avoid collisions, the serial interface (SI) routing must be separated to ensure that only one TDM accesses the UCC at any given time.

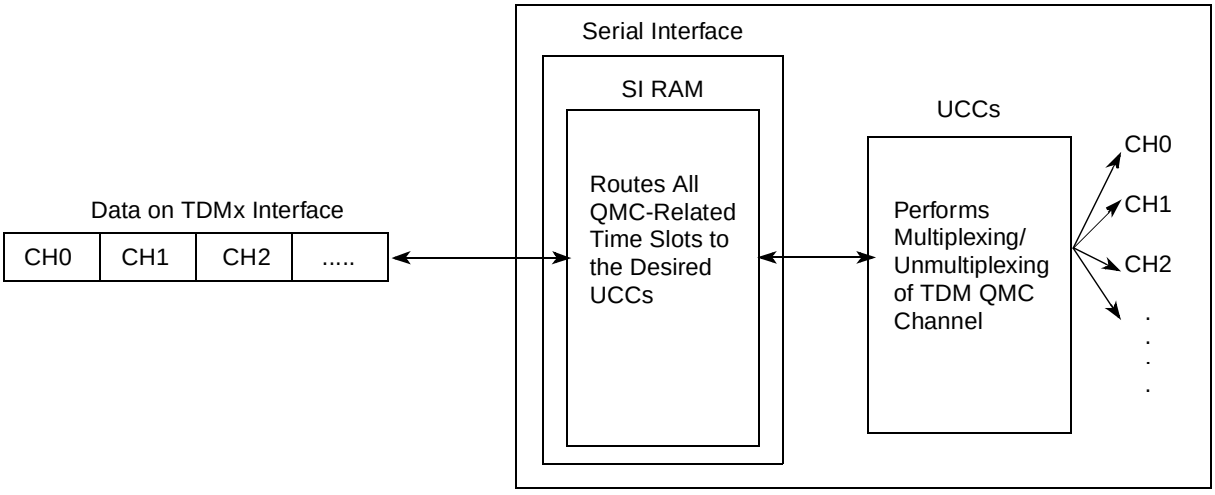


Figure 34-1. QMC Channel Addressing Capability

34.1 Features

QMC-specific features include the following:

- Up to 64 independent communication channels per UCC
- Arbitrary mapping of any of 0–63 channels to any TDM time slot
- 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
- Subchannel masking 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
- H0, H11, and H12 ISDN channels
- 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 multi-user RAM for parameter storage

34.2 QMC and the Serial Interface

This section describes additional functionality that the SI provides to QMC operation. The QMC works in conjunction with the serial interface, taking advantage of its programmable SIRAM and additional functionality. See Chapter 32, “Serial Interface with Time-Slot Assigner,” for details on proper programming of the SI registers and SIRAM. However, the QMC can operate in non-multiplexed serial interface (NMSI) mode, directly using the UCC pins instead of the TDM interface pins. Functions such as frame synchronization, loopback, echo, and inverted signals are performed in the serial interface and cannot be achieved in NMSI mode. Use of the serial interface is recommended even if only one UCC is used for the TDM bus.

To connect a UCC to the SI or to its own pins in NMSI mode, program the appropriate CMX UCC clock route register (CMXUCRx). See Section 20.5.4, “CMX UCC Clock Route Register (CMXUCR1).”

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 is 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. 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 UCC 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 GUMR_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.

NOTE

The number of slots mapped in the SI RAM to the UCC must remain constant while QMC is activated. Routing changes may only be made within the QMC routing tables.

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 multi-user RAM to access data within the buffers.

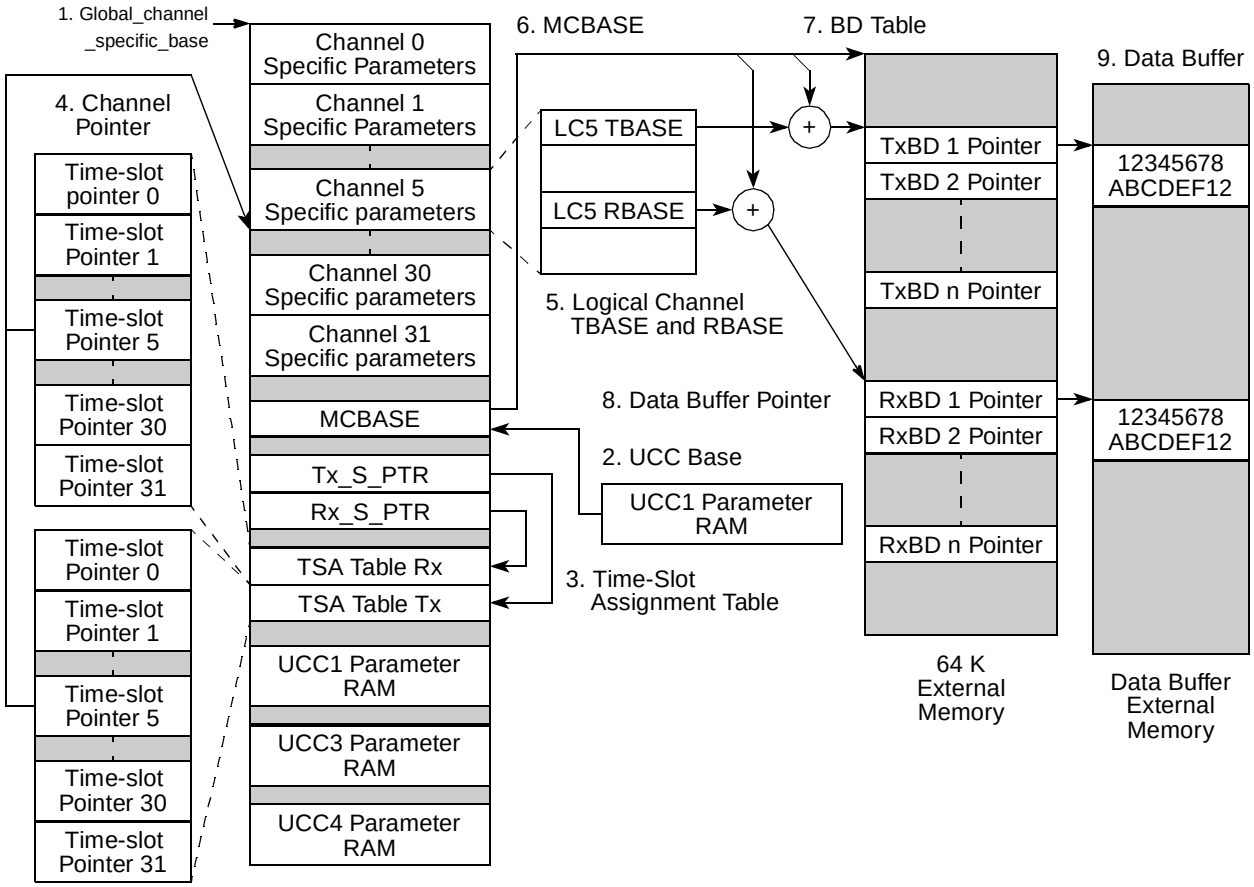


Figure 34-2. QMC Memory Structure

34.3.2 UCC Base and Global Multichannel Parameters

The UCC base points to the start of the parameter RAM for each UCC. When the QMC protocol is enabled on a UCC, its parameter RAM is used to store the global multichannel parameters for all the logical channels. This area contains parameters and pointers that are common to all channels.

34.3.2.1 TSATRx/TSATTx Pointers and Time-Slot Assignment Table

The time-slot assignment table pointers are within the global multichannel parameters. There are two pointers—Tx_S_PTR for transmit and Rx_S_PTR for receive. The Rx_S_PTR is normally set to UCC Base + 0x20; this is the normal location of the receive time slot assignment table. The Tx_S_PTR is normally set to UCC Base + 0x60; 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 UCC base + 0x20 so that Rx and Tx have a common time-slot assignment table. See Section 34.3.3, "Multiple UCC Assignment Tables," for more information. The time-slot assignment table holds one 16-bit entry for each time slot. It has options for subchannel masking, 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 must be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another UCC. Associated with the Rx/Tx_S_PTR are the Rx/TxPTR pointers that are maintained by the QUICC Engine block 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 multi-user 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 multi-user RAM is required (32×64), and for 64-channel support, 4 Kbytes of multi-user RAM is required (64×64). In most cases, time slot 0 channel pointer addresses the base of multi-user RAM for logical channel 0, and time slot 1 channel pointer would address the base of multi-user 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

Multiple time slots can be concatenated to one logical channel 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 multichannel parameters. Each UCC has a unique MCBASE value pointing to the base of the UCC'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 MURAM. This may save valuable access time if external memory is slow.

34.3.2.5 Buffer Descriptor Table

A buffer descriptor table for each UCC 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 UCC has a maximum of 8,192 (64 Kbytes

memory ÷ 8-byte pointers) buffers. For a 32-channel implementation, each logical channel has a maximum of 128(8,192 / (32 × 2)) buffers for receive and 128 buffers for transmit. If any BDs are placed in internal memory, the GBL bit of RSTATE and TSTATE may not be set. For each logical channel, a circular queue exists with programmable start address and length.

34.3.2.6 Data Buffer Pointer

As with the standard QUICC Engine 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 Multichannel Parameters

The global multichannel parameters reside in the UCC'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 UCC. 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 UCCs, this table only needs to be present once for multiple UCCs operating in QMC mode. See Section 34.3.3, "Multiple UCC Assignment Tables," for more information. For 64-channel support there is only space for one table; therefore common Rx and Tx parameters must be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another UCC.

The multi-user RAM is used for the channel-specific area for all UCCs. It is important that individual time slots are mapped to only one UCC, 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 Multichannel Parameters

Offset to UCC Base	Name	Width (Bits)	Description
0x00	**MCBASE**	32	Multichannel 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.
0x04	**QMCSTATE**	16	Multichannel controller state (initialize to 0x8000)—Internal QMC state machine value used by RISC processor for global state definition.
0x06	**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 value must be a multiple of 4 bytes.

Table 34-1. Global Multichannel Parameters (continued)

Offset to UCC Base	Name	Width (Bits)	Description
0x08	**Tx_S_PTR**	16	Tx time-slot assignment table pointer (UCC base + 0x60 in normal mode; UCC base + 0x20 for common Rx and Tx time slot assignment tables)—This global QMC parameter defines the start value of the TSATTx table. The TSATTx table in the global multichannel parameter listing starts by default at UCC base + 0x60. 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 UCC base + 0x20. This parameter is an offset from multi-user RAM BASE. This table must be present only once per UCC global area. Other UCCs can access this location.
0x0A	**RxPTR**	16	Rx pointer (initialize to UCC base + 0x20)—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.
0x0C	**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.”
0x0E	**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 QUICC Engine block after a global interrupt.
0x10	**INTBASE**	32	Multichannel 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 UCC operating in QMC mode has its own interrupt table in external memory. See Section 34.5, “QMC Exceptions.”
0x14	**INTPTR**	32	Multichannel 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.
0x18	**Rx_S_PTR**	16	Rx time-slot assignment table pointer (default = UCC base + 0x20 in normal mode)—This global QMC parameter defines the start value of the TSATRx table, which must be present only once per UCC global area. Other UCCs may access this location.
0x1A	**TxPTR**	16	TxPTR (initialize to UCC Base + 0x60)—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.

Table 34-1. Global Multichannel Parameters (continued)

Offset to UCC Base	Name	Width (Bits)	Description
0x1C	**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).
0x20	**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 UCC. 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 UCC and to do all enabling and masking in the time-slot assignment table. See Figure 34-4.
0x60	**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 UCC 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 UCC and to do all enabling and masking in the time slot assignment table. See Figure 34-4.
0xA0	**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).
0xA2	Reserved	16	—
0xA4	TEMP_RBA	32	Temporary receive buffer address
0xA8	TEMP_CRC	32	Temporary cyclic redundancy check
0xAC	RX_FRM_Base	16	This entry contains a pointer to an area in the multi-user RAM where the receiver framer temporary parameters reside. The area designated for these parameters is a single byte per TDM channel.
0xAE	TX_FRM_Base	16	This entry contains a pointer to an area in the multi-user RAM where the receiver framer temporary parameters reside. The area designated for these parameters is a single byte per TDM channel.
0xB0–0xC3	Reserved	—	Should be initialized to zero
0xC4	**QMC_Global_Channel_specific_base**	32	QMC Global Channel Specific Parameters' Base. Internal pointer for the new channel base address. Must be cleared (Set to 0) if Channel Specific Parameters reside at the beginning of the MURAM.

NOTE

The receiver and transmitter cannot share the same structure of RX_FRM_Base as done on the QMC time-slot assignment (TSA) table when the TDM functionality of the receiver is identical to that of the transmitter.

The user must program the separate areas pointed by RX_FRM_Base/TX_FRM_Base so that 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 mapped QMC channels in the system and is indexed using the channel number. Each time a channel is initialized, the corresponding entry should be programmed to this value.

0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	1

Figure 34-3. RX Framer Entry

NOTE

The area between UCC base + 0x20 and UCC base + 0x9F is normally used for TSA tables. The preceding mapping is ideal for 32-channel support. The exact mapping of the TSA tables is determined by the configuration of Rx_S_PTR and Tx_S_PTR and is not fixed. For 64-channel support, common Rx and Tx parameters are recommended. The TSA table is common and has 64 entries starting at UCC base + 0x20; see Figure 34-5. Alternatively, another UCC parameter RAM can be used, as determined by Rx_S_PTR and Tx_S_PTR; see Figure 34-7. However implemented, the TSA tables can reside anywhere in internal memory.

Figure 34-4 shows a general QMC time-slot assignment table for 32 16-bit time slots. The fields are used either to transmit or receive channels.

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)
	...				32 × 16
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 30	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)
Time Slot 31	V	W	Mask(0:1)	Channel Pointer	Mask(2:7)

Figure 34-4. QMC 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—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 QUICC Engine block 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—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—Identifies the last entry in TSATTx. 0 This is not the last time slot in the frame. 1 The QUICC Engine block wraps around and handles time slot 0 or the first 8 bits of data in the UCC 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, a common time-slot assignment table can be used. It is initialized by setting both Tx_S_PTR and Rx_S_PTR to UCC Base + 0x20. For 64-channel support, common Rx and Tx parameters are recommended. The time-slot assignment table is then also common and has 64 entries starting at UCC Base + 0x20; see Figure 34-5.

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 × 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 34-5. Time Slot Assignment Table for 64-Channel Common Rx and Tx Mapping

34.3.3 Multiple UCC Assignment Tables

Assume a scenario as depicted in Figure 34-6 as an artificial example for a complex setup. A 2.048-Mbps TDM link is fed directly into the TSA. Within the SI RAM, the even channels (byte-wide) are multiplexed to UCC3 and the odd channels are multiplexed to UCC1. Each UCC sees a continuous byte stream without any gaps as described earlier.

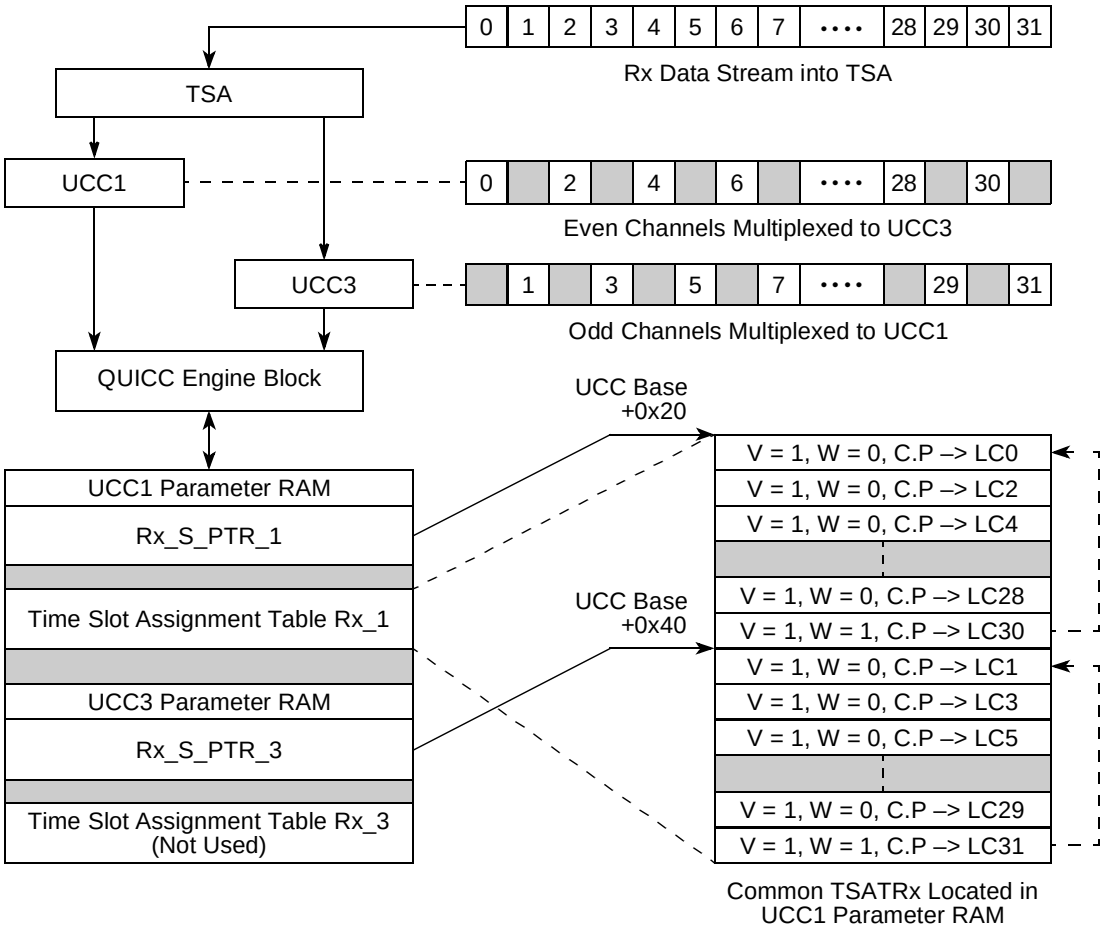


Figure 34-6. Rx Time Slot Assignment Table for 32 Channels over Two UCCs

NOTE

Route multiples of bytes to each UCC to delineate between time slots. Unused bits are routed to the UCC and masked in the time-slot assignment table.

In Figure 34-6, each UCC has its own pointer, Rx_S_PTR_1 and Rx_S_PTR_3, addressing the UCC1 time-slot assignment table. This table must be present only once in one of the UCC1 global parameter areas. Rx_S_PTR_1 points to the start of the table, address UCC base + 0x20. The 16 logical channels from UCC1 are located in the first 16 entries of the table. The entry for logical channel 30 has the wrap bit (W) set, causing the QUICC Engine block to wrap back to logical channel 0 on reception of the next byte routed to UCC1. Rx_S_PTR_3 addresses UCC base + 0x40, the start of the 16 entries for UCC3. The entry for logical channel 31 has the wrap bit (W) set, causing the QUICC Engine block to wrap back to logical

channel 1 on reception of the next byte routed to UCC3. Each entry within the table has a channel pointer to a logical channel. It is important that different UCCs do not point to the same logical channel. The TSATTx is also located in UCC1 parameter RAM, so the area reserved for the TSA tables in UCC3 parameter RAM is free for alternative use.

A second scenario is depicted in Figure 34-7. A 4.096-Mbps TDM link is fed directly into the TSA. Again, within the SI RAM, the even channels (byte-wide) are multiplexed to UCC3 and the odd channels are multiplexed to UCC1. This requires two 64-entry tables with 256 bytes total, but only 128 bytes are allocated in the parameter RAM of an UCC for time-slot assignment tables. In this case, the Rx table is located in UCC1 parameter RAM, and the TX table is located in UCC3 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.

There can be a time-slot assignment table for every UCC in its corresponding RAM page and with all the TDM routed to the different UCCs. The result is a very flexible system that can be changed on-the-fly without disconnecting the TDM interface.

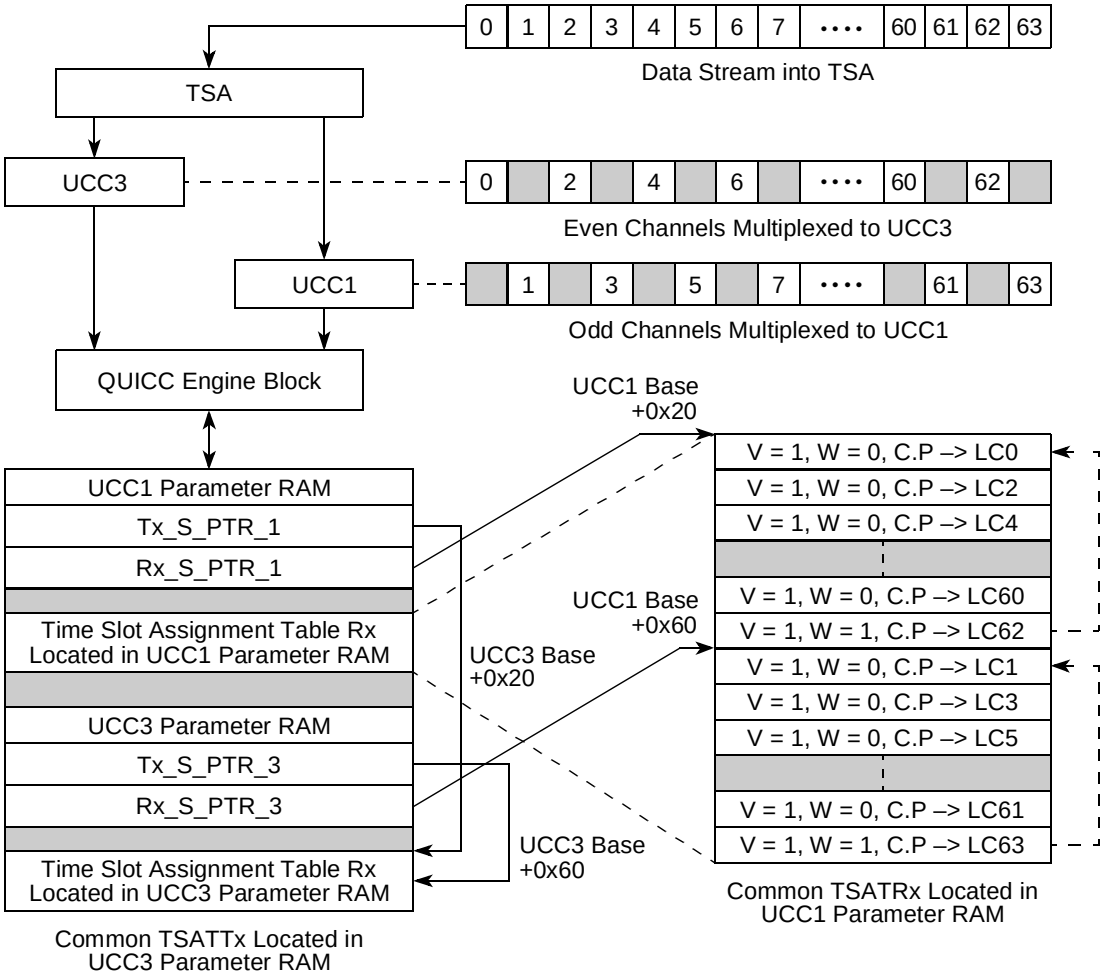


Figure 34-7. Time-Slot Assignment Tables for 64 Channels over 2 UCCs

34.3.4 Channel Specific Parameters

The channel-specific parameters are located in the lower part of the multi-user 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 UCCs. 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
0x00	**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.
0x02	**CHAMR**	16	Channel mode register. See Section 34.3.4.1.1, "CHAMR—Channel Mode Register (HDLC)."
0x04	**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)."
0x08	—	32	Tx internal data pointer—Points to current absolute address of channel.
0x0C	**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.
0x0E	—	16	Tx internal byte count—Number of remaining bytes
0x10	TUPACK	32	(Tx Temp) Unpack 4 bytes from 1 long word
0x14	**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.
0x18	TCRC	32	Temp transmit CRC—Temp value of CRC calculation result
0x1C	**INTMSK**	16	Channel's interrupt mask flags—See Section 34.3.4.1.3, "INTMSK—Interrupt Mask (HDLC)."
0x1E	BDFlags	16	Temp
0x20	**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.
0x22	**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 are received. At this point, the SDMA transfers the long word to memory, and all 8 bytes are in the receive buffer. Also, the MFLR violation (>5) is detected and the interrupt may be generated. No more data is written into this buffer when the MFLR violation is detected.
0x24	**RSTATE**	32	Rx internal state. See Section 34.3.4.1.4, "RSTATE—Rx Internal State (HDLC)," for more information.
0x28	—	32	Rx internal data pointer—Points to current address of specific channel.
0x2C	**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.
0x2E	—	16	Rx internal byte count—Per Channel: Number of remaining bytes in buffer

Table 34-4. Channel-Specific HDLC Parameters (continued)

Offset	Name	Width (Bits)	Description
0x30	**RPACK**	32	(Rx Temp) Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0x8000_0000.
0x34	**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.
0x38	RCRC	32	Temp receive CRC—Temp value of CRC calculation result
0x3C	MAX_cnt	16	Max_length counter—Count length remaining
0x3E	TMP_MB	16	Temp—Holds MIN(MAX_cnt, Rx internal byte count)

34.3.4.1.1 CHAMR—Channel Mode Register (HDLC)

CHAMR is a host-initialized register. Figure 34-8 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 34-8. 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)

Field	Name	Description
0	MODE	Mode. Each channel has a programmable option to use transparent mode or HDLC mode. 0 Transparent mode 1 HDLC mode
1	—	Reserved, should be cleared.
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 QUICC Engine 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 initial value before the idle is 1s. In this case, it is assumed the transmitter is uninitialized. An uninitialized UCC transmits 1s in every position.

Table 34-5. CHAMR Field Descriptions (HDLC) (continued)

Field	Name	Description
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 34-4) 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, should be cleared.
7	**POL**	Enable polling. This bit enables the transmitter to poll the transmit buffer descriptors. 0 The QUICC Engine block does not check the ready bit (R) in the transmit buffer descriptor. 1 The QUICC Engine block 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 multiframe 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 QUICC Engine block; the user should not attempt to clear this bit in software.
8	**CRC**	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	—	Reserved, should be cleared.
10–11	—	Reserved, should be cleared.
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. Figure 34-9 shows the TSTATE register for HDLC operation.

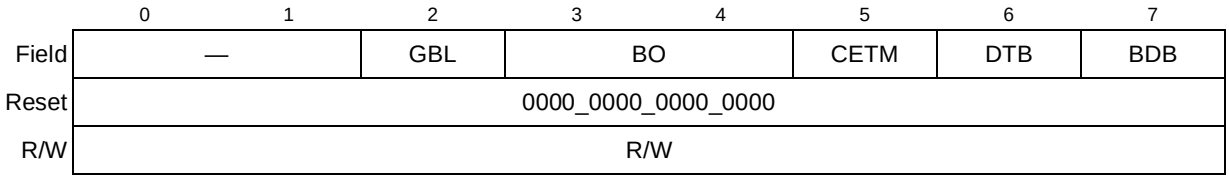


Figure 34-9. TSTATE—TX Internal State (HDLC)

Table 34-6 describes the 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 To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. 00, 01, 11Reserved 10 Big endian
5	**CETM**	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the “Debug Configuration” subsection in your device reference manual.
6	DTB	Should be cleared
7	BDB	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 34-11. 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 cleared to zero. Refer to Section 34.5, “QMC Exceptions.”

0 = No interrupt request is generated and no new entry is written in the circular interrupt table.

1 = Interrupts are enabled.

The host initializes this register 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 34-10. 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 34-11. INTMSK (HDLC)

34.3.4.1.4 RSTATE—Rx Internal State (HDLC)

RSTATE is host-initialized before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command. Figure 34-12 shows the RSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	DTB	BDB
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 34-12. RSTATE—RX Internal State (HDLC)

Table 34-7 describes the 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 To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
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, 01, 11Reserved 1x Big endian
5	**CETM**	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the “Debug Configuration” subsection in your device manual.”
6	DTB	Should be cleared
7	BDB	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
0x00	**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.
0x02	**CHAMR**	16	Channel mode register. See Section 34.3.4.2.1, “CHAMR—Channel Mode Register (Transparent Mode).”

Table 34-8. Channel-Specific Transparent Parameters (continued)

Offset	Name	Width	Description
0x04	**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)."
0x08	—	32	Tx internal data pointer—Points to current absolute address of channel.
0x0C	**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.
0x0E	—	16	Tx internal byte count—Number of remaining bytes
0x10	TUPACK	32	(Tx temp) Unpack 4 bytes from 1 long word
0x14	**ZISTATE**	32	Zero-insertion machine state (host-initialized to 0x0000_0200)—Contains the previous state of the zero-insertion state machine.
0x18	RES	32	—
0x1C	**INTMSK**	16	Channel's interrupt mask flags. See Figure 34-16.
0x1E	BDFlags	16	Temp
0x20	**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.
0x22	**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.
0x24	**RSTATE**	32	Rx internal state. See Section 34.3.4.2.5, "RSTATE—Rx Internal State (Transparent Mode)," for more information.
0x28	—	32	Rx internal data pointer—Points to current address of specific channel.
0x2C	**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.
0x2E	—	16	Rx internal byte count—Per channel: Number of remaining bytes in buffer
0x30	RPACK	32	(Rx temp) Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0x8000_0000.
0x34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x003F_FFE2 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.
0x38	RES	32	—
0x3C	**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."
0x3E	RES	16	—

34.3.4.2.1 CHAMR—Channel Mode Register (Transparent Mode)

CHAMR is a host-initialized register. Figure 34-13 shows the channel mode register for transparent mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	15
Field	MODE	RD	1	ENT	—	SYNC	—	POL	0	0	—		0	
Reset	0000_0000_0000_0000													

Figure 34-13. CHAMR—Channel Mode Register (Transparent Mode)

Table 34-9 describes the channel mode register fields for transparent operation. Boldfaced parameters must be initialized by the user.

Table 34-9. CHAMR Bit Settings (Transparent Mode)

Field	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 is not 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	—	Set to 1
3	**ENT**	Enable transmit 0 Disable transmitter. If this bit is cleared and the channel transmitter is routed to a certain time slot (within TSATTx, see Figure 34-4) the transmitter sends 1s 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, should be cleared.
5	**SYNC**	Synchronization. Controls synchronization of multichannel 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	—	Reserved, should be cleared.
7	**POL**	Enable polling. Enables the transmitter to poll the transmit BDs. 0 The QUICC Engine block does not check the ready (R) bit in the transmit buffer descriptor. 1 The QUICC Engine block checks 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 when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared, that is, at the end of a frame or at the end of a multiframe transmission. To prevent deadlock, software should prepare the new BD, or multiple BDs, and set the ready (R) bit in the BD before setting POL. Note that the QUICC Engine block automatically clears this bit; software should never try to clear this bit.
8–9	—	Reserved, should be cleared.
10–11	—	Reserved, should be cleared.
12–15	—	Reserved, should be cleared.

34.3.4.2.2 TSTATE—Tx Internal State (Transparent Mode)

TSTATE defines the internal transmitter state. Figure 34-14 shows the TSTATE register for transparent mode.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 34-14. TSTATE—TX Internal State (Transparent Mode)

Table 34-10 describes the 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 To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. 00 , 01, 11 Reserved 10 Big endian
5	**CETM**	QUICC Engine Transaction Mark. The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block for debug purposes. See the “Debug Configuration” subsection in your device manual.
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 34-16. 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 cleared.

0 = No interrupt request is generated and no new entry is written in the circular interrupt table.

1 = Interrupts are enabled.

The host initializes this register 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 34-15. Interrupt Table Entry

	0	10	11	15
Field	Reserved		Interrupt Mask Bits	
Reset	0000_0000_0000_0000			

Figure 34-16. INTMSK (Transparent Mode)

34.3.4.2.4 TRNSYNC—Transparent Synchronization

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.

When a transparent message is sent over several time slots, the time slot in which the first byte of data appear is to appear must be known.

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:

TSn, TSn + 1, TSn + 2 ..... TSn + x

The algorithm for the receiver byte in decimal is:

$(TSn + 1) \times 2$

The algorithm for the transmit byte in decimal is:

$(TSn + x + 1) \times 2$

The result from these calculations is a decimal value programmed into TRNSYNC.

NOTE

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 can 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 34-17, 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 UCC 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 TSn = 7
  - Rx Byte:$(7+1) \times 2 = 16$
  - As x = 0, TSn + x = TSn = 7, so Tx Byte:$(7 + 1) \times 2 = 16$
- C2 is a single byte in TS23, so TSn = 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, TSn + x = TSn = 23, so Tx Byte:(23 + 1) × 2 = 0
- C3 is a 2-byte pattern TS7, TS8, so TSn = 7
  - Rx Byte: (7 + 1) × 2 = 16
  - As x = 1, TSn + x = 8, so Tx Byte:(8 + 1) × 2 = 18
- C4 is a 2-byte pattern TS8, TS7, so TSn = 8
  - Rx Byte:(8+1) × 2 = 16
  - As x = 1, TSn + x = 7, so Tx Byte:(7 + 1) × 2 = 16
- C5 is a 2-byte pattern TS19, TS23, so TSn = 19
  - Rx Byte:(19 + 1) × 2 = 40
  - As x = 1, TSn + x = 23, so Tx Byte:(23 + 1) × 2 = 0
- C6 is a 2-byte pattern TS23, TS19, so TSn = 23
  - Rx Byte:(23 + 1) × 2 = 0
  - As x = 1, TSn + x = 19, so Tx Byte:(19 + 1) × 2 = 40
- C7 is a 4-byte pattern TS20, TS23, TS8, TS9, and TS19, so TSn = 20
  - Rx Byte:(20 + 1) × 2 = 42
  - As x = 5, TSn + x = 19, so Tx Byte:(19 + 1) × 2 = 40
- C8 is a 4-byte pattern TS8, TS9, TS19, TS20, and TS23, so TSn = 8
  - Rx Byte:(8 + 1) × 2 = 18
  - As x = 5, TSn + x = 23, so Tx Byte:(23 + 1) × 2 = 0
- C9 is a 4-byte pattern TS19, TS20, TS23, TS8, and TS9, so TSn = 19
  - Rx Byte:(19 + 1) × 2 = 40
  - As x = 5, TSn + x = 9, so Tx Byte:(9 + 1) × 2 = 20

NOTE

Case C1 and C2 can be used as described here. A more elegant solution for single time-slot applications is to have the SYNC bit cleared in the channel mode register. Both scenarios produce the same result.

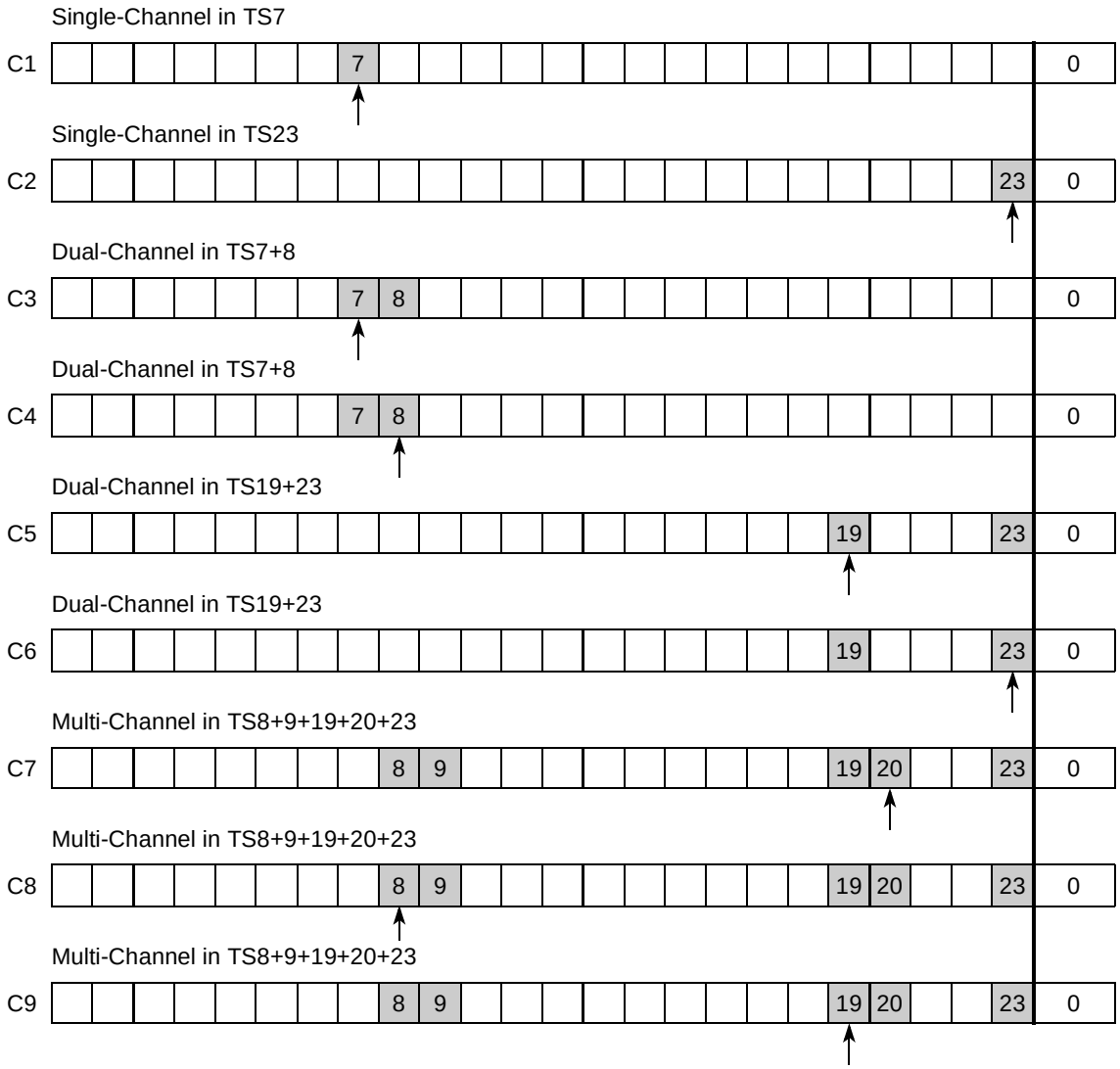


Figure 34-17. Examples of Different T1 Time Slot Allocation

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.

Figure 34-18 shows the RSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 34-18. RSTATE—RX Internal State (Transparent)

Table 34-11 describes the RSTATE fields.

Table 34-11. RSTATE Field Descriptions (Transparent)

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. 00, 01, 11 Reserved 10 Big endian
5	**CETM**	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine block, for debug purposes. See the “Debug Configuration” subsection in your device manual.”
6–7	—	Reserved, should be cleared.

34.4 QMC Commands

The host issues commands to the QMC by writing to the command register (CECR); see Section 19.3.1, “QUICC Engine Command Register (CECR).”

34.4.1 Transmit Commands

The stop transmit command, shown as follows, disables the transmission of data on the selected channel and clears CHAMR[POL].

```
STOP TRANSMIT <channel>
```

When this command is issued in the middle of a frame, the RISC processor sends an ABORT indication (7F) on the selected channel followed by IDLEs or FLAGs, depending on the mode. If this command is issued between frames, the RISC processor continues sending IDLEs or FLAGs in this channel (depending on CHAMR[IDLM]). The Tx buffer descriptor pointer (TBPTR) is not advanced to the next buffer; see Table 34-4 and Section 34.3.2.8, “Global Multichannel Parameters.”

Set the CHAMR[POL] bit t 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

The stop receive command, shown as follows, forces the receiver of the selected channel to stop receiving.

```
STOP RECEIVE<channel>
```

After this command is issued, the microcode does not change any receive parameters in the multi-user RAM. The command immediately stops activity on the channel. It does not wait for a frame reception to finish. Because the current buffer descriptor can remain open if a reception was in progress, errors may appear on this buffer when reception restarts. Initialize ZDSTATE and then RSTATE to their initial values to start reception or to continue receiving after a stop command.

NOTE

There is no command to start transmission and reception. This function is performed through the GUMR register.

34.5 QMC Exceptions

QMC interrupt handling involves two principle data structures—the UCC event register (UCCE) and the circular interrupt table. Figure 34-19 illustrates the circular interrupt table.

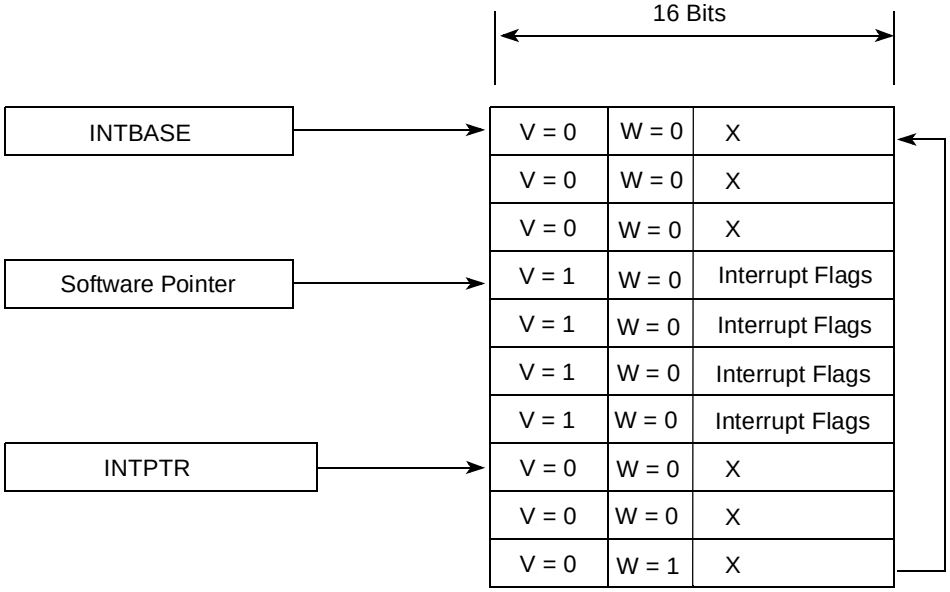


Figure 34-19. Circular Interrupt Table in External Memory

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 UCC event register (UCCE) according to the type of exception.

Following a request that is not masked out by the INTMASK or the UCCM (UCC mask) register, an interrupt is generated to the host. The host reads the UCCE to determine the cause of interrupt. A dedicated UCCE 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) and all event bits in the entry. 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 clears all bits but the wrap bit in a queue entry after reading it and before handling it, even though its valid bit may be cleared. Clearing only the valid bits confuses user software with incorrect channel interrupts from left over set bits.

34.5.1 Global Error Events

A global error affects the operation of the UCC. A global error can occur for two reasons— serial data rates being too high for the QUICC Engine block to handle, and QUICC Engine 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 QUICC Engine block is governed by transfers between the UCC and the SI. Every transfer in either direction causes a request to the QUICC Engine state machine. If requests are received too quickly, the QUICC Engine block may lose track of the mapping of serial data to the QMC channels. If this happens, the QUICC Engine block 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 the 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 1s in each time slot.
- Generates an interrupt request to the host (if enabled) and sets the GUN bit in the UCCE 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 UCC. 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 UCCE.
- Stops writing data to all channels' 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 UCC Event Register (UCCE)

The QMC's UCCE is used to report events and generate interrupt requests. See Figure 34-20 and Table 34-12 for UCCE field descriptions. For each of its flags, a corresponding programmable mask/enable bit in the UCCM determines whether an interrupt request is generated. If a bit in the UCCM register is zero, the corresponding interrupt flag does not survive, and the QUICC Engine block does not proceed with its usual interrupt handling. If a bit in the UCCM is set, the corresponding interrupt flag in the UCCE survives, and the UCC event bit is set in the QUICC Engine interrupt-pending register. See Figure 34-21 for UCCM assignments.

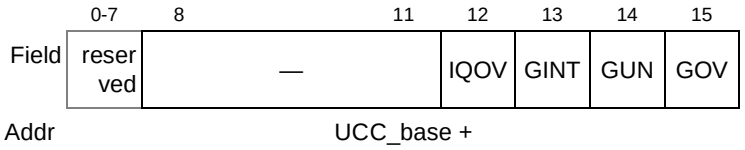


Figure 34-20. UCC Event Register

Table 34-12. UCC Event Register Field Descriptions

Field	Name	Description
0–11	—	Reserved
12	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 UCCE and recognizing this condition by writing a 1 to its location in the UCCE. 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.
13	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 UCCE. 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.
14	GUN	Global transmitter underrun 0 No global transmitter underrun has occurred. 1 This flag indicates that an underrun occurred in the UCC'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 UCCE, 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 UCCM, an interrupt request is generated when GUN is set. The host may clear GUN by writing 1 to its location in the UCCE.
15	GOV	Global receiver overrun 0 No global receiver overrun has occurred. 1 This flag indicates that an overrun occurred in the UCC'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 UCCE, 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 UCCM, an interrupt request is generated when GOV is set. The host may clear GOV by writing 1 to its location in the UCCE.

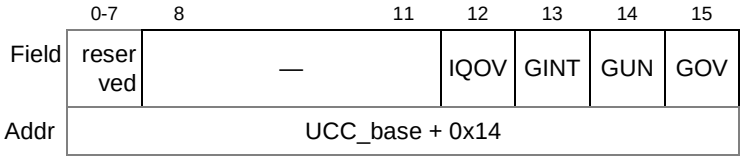


Figure 34-21. UCCM Register

34.5.3 Interrupt Table Entry

The interrupt table contains information about channel-specific events. Its flags are shown in Figure 34-22. 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.

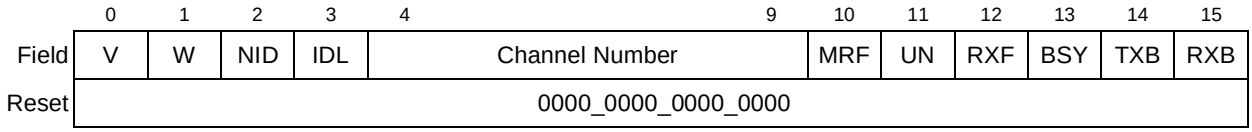


Figure 34-22. Interrupt Table Entry

Table 34-13. Interrupt Table Entry Field Descriptions

Field	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)

Field	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 multibuffer transmission already in progress. Transmission of a frame is a continuous bit stream 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 UCC'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 UCCE 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 software, clear the recognized interrupt bits in the UCCE before actually handling the interrupt entries. Clear only the bits being handled. Never clear bits for events in the UCCE that are not handled. This facilitates debugging.

For 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 them, these bits confuse the software when the entry is used again. QMC only ORs in new bits and numbers. It does not overwrite and clear previously set bits.

34.5.5 Channel Interrupt Processing Flow

Figure 34-23 illustrates the flow of how the QUICC Engine block generates a channel interrupt. Note that this does not describe the processing of the global interrupts GUN and GOV.

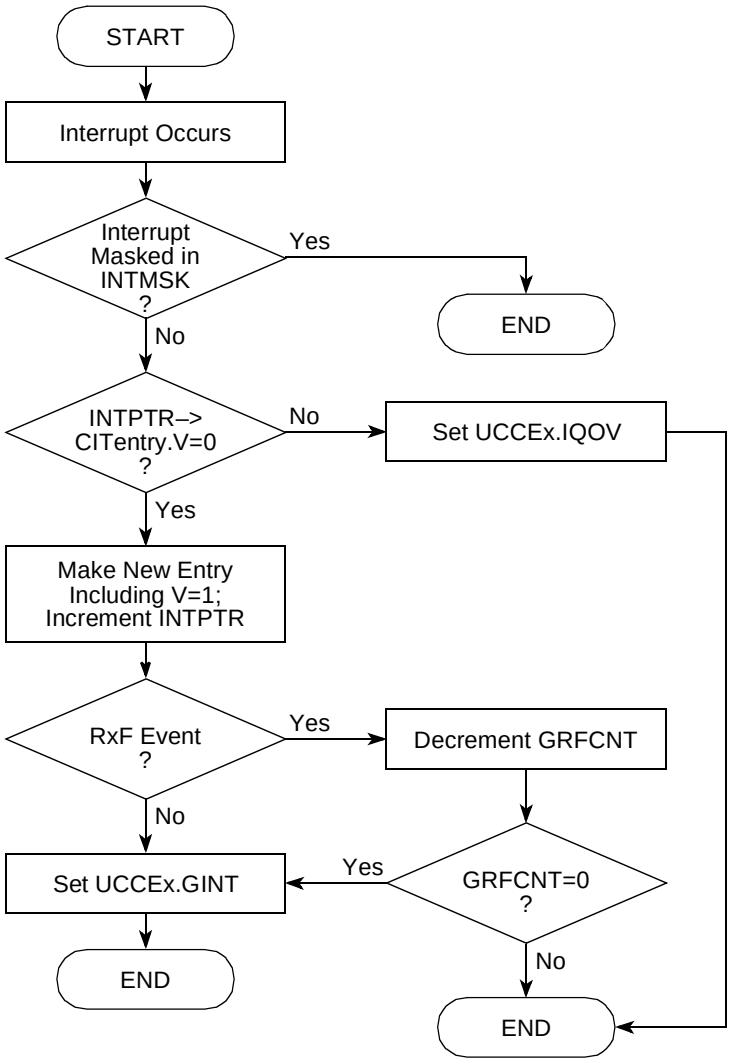


Figure 34-23. Channel Interrupt Generation Flow

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.

Section 34.6.1, “Receive Buffer Descriptor,” and Section 34.6.2, “Transmit Buffer Descriptor,” describe the contents of the receive and transmit buffer descriptors for the QMC protocol.

Section 34.6.3, “Placement of Buffer Descriptors,” discusses the placement of QMC and non-QMC buffer descriptors in internal and external memory.

NOTE

The QUICC Engine block ORs the various status bits into the BD. Clear all the status bits generated by the QUICC Engine block before (re-)enabling the BD, or you may confuse your own software with leftover old status values.

34.6.1 Receive Buffer Descriptor

Figure 34-24 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																

Notes: Entries in boldface must be initialized by the user.

Figure 34-24. 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).
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.

Table 34-14. RxBD Field Descriptions (continued)

Bits	Name	Description
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 QUICC Engine block 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, the CP may not mark the BD that contains the last of the frame data as the last BD in the frame. The CP closes the BD, marks it with E = 0, sets the Data Length field to the length of the data in this buffer, but leaves L (bit 4) cleared. Then, the CP closes the next BD, erroneously marks it as containing data (E = 0), and marks it as the last BD in the frame (L = 1).

Figure 34-25 shows how nonoctet 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.

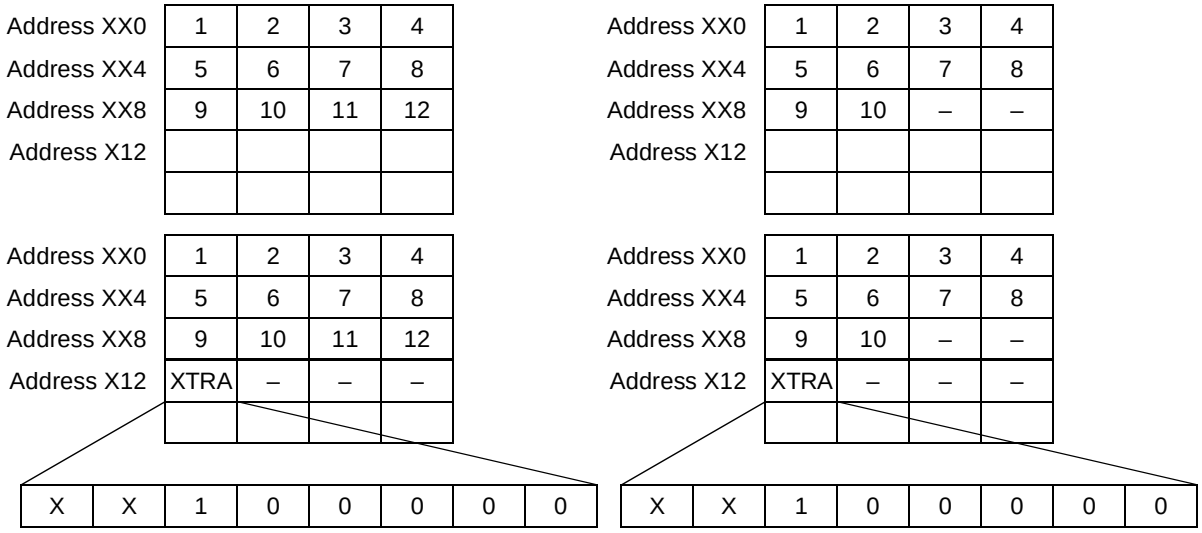


Figure 34-25. Nonoctet Alignment Data

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 nonoctet 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.

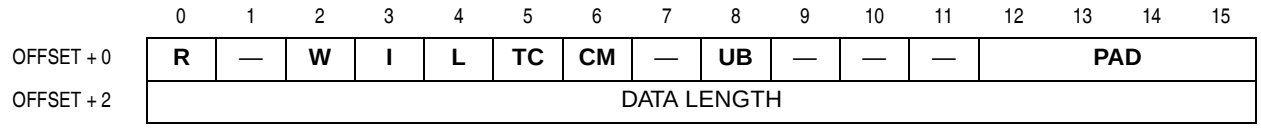
The two diagrams on the right show how the data and the extra information is stored for a frame length that is not a multiple of 4 bytes. The additional information is always on a long-word boundary. In the top case the frame length is 10 bytes and in the bottom case the length is 11 bytes.

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 34-25 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 34-26 shows the transmit buffer descriptor.



OFFSET + 4 OFFSET + 6	Tx DATA BUFFER POINTER

Notes: Entries in boldface must be initialized by the user.

Figure 34-26. 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

Field	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 QUICC Engine block 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 QUICC Engine block 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 multi-user 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. **Note:** In Transparent Mode operation, setting Last puts the channel to Idle state after sending the last byte of the buffer (or the CRC, if enabled). To resume transmit, set the POL bit in the CHAMR. For continuous transmission—for example, with no concept of a frame boundary—Last should NOT be set in Transparent Mode.
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 QUICC Engine block after this buffer descriptor is closed, allowing the associated data buffer to be retransmitted automatically when the QUICC Engine block next accesses this buffer descriptor.
7	—	—
8	**UB**	User bit. The QUICC Engine block 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	—	—

Table 34-15. Transmit Buffer Descriptor (TxBD) Field Descriptions (continued)

Field	Name	Description
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 UCC1 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 is generated; see Figure 34-27.
16–31	**DL**	Data length. The data length is the number of bytes the QUICC Engine block should transmit from this buffer descriptor's data buffer. It is never modified by the QUICC Engine block. 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 QUICC Engine block.

Figure 34-27 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.

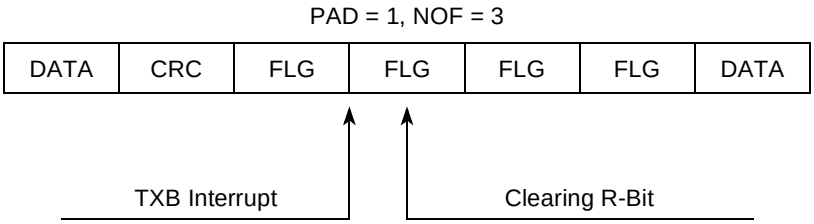


Figure 34-27. Relation between PAD and NOF

34.6.3 Placement of Buffer Descriptors

The internal multi-user 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 QUICC Engine block 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 UCCs

There are two possible memory configurations for operating the QMC protocol on several UCCs. For each UCC 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 UCC in the parameter RAM can be duplicated, requiring 190 bytes per UCC.

Alternatively, multiple UCCs 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 UCC RAM page uses 190 bytes, 62 bytes for parameter fields and 128 bytes for the common time slot assignment tables, allowing the other UCCs 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 internal memory, refer to the "Memory Map" chapter of your device reference manual.

To support 32 channels, only 2-Kbyte multi-user 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 multi-user RAM for buffer descriptors for other protocols.

To support 64 channels, 4-Kbyte multi-user 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 multi-user RAM. Each unused logical channel creates a 64-byte hole in the multi-user RAM. This area is then free for any other use.

34.7 QMC Initialization

The following are the general initialization steps necessary after a reset for QMC operation. Prior to completing these steps, ensure that the values of CECR[OPCODE], see Section 19.3.1, "QUICC Engine Command Register (CECR)," and GUMR_L[MODE], see Figure 23-1," are set 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 TxBDs and corresponding Tx data buffers
4. Initialize RxBDs
5. Connect the UCC to the TDM interface through the appropriate UCC clock multiplexing register CMXUCR*x*
6. Perform a re-initialization sequence as described in Section 34.8, "QMC Re-Initialization," to ensure a correct UCC setup in all scenarios.
7. Initialize UCC registers, including enabling the UCC. Program GUMR_H[16] to a zero to select QMC mode, see Table 23-3.
8. Program RX and TX SIRAM to point appropriate time slots to the UCC used for QMC.
9. Initialize SI registers

10. Enable TDM

34.7.1 Restarting the Transmitter

A global underrun may require the UCC 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 UCC 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 continues 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.

34.8 QMC Re-Initialization

If the QMC is disabled and enabled in a re-configuration or re-initialization procedure, the QMC controller must be re-initialized.

Before enabling the UCC used for the QMC controller (set of GUMR_L[ENT] or GUMR_L[ENR]):

1. Re-initialize the QMC Tx or Rx startup addresses with the PUSHSCHED host command (see Table 34-16 and Example 34-1).
2. Re-initialize RxPTR or TxPTR in QMC global parameter RAM to Rx_S_PTR or Tx_S_PTR, respectively.

Table 34-16. PUSHSCHED Host Command

	Tx	Rx
PUSHSCHED Address in CECDR	0x80	0x82

Example 34-1 shows a QMC re-initialization example.

Example 34-1. QMC Re-Initialization Example

```
#define QE_CR_SUBBLOCK_UCCSLOW1     0x00000000
#define QE_CR_SUBBLOCK_UCCSLOW2     0x00200000
#define QE_CR_SUBBLOCK_UCCSLOW3     0x00400000
#define QE_CR_SUBBLOCK_UCCSLOW4     0x00600000
#define QE_CR_SUBBLOCK_UCCSLOW5     0x00800000
#define QE_CR_SUBBLOCK_UCCSLOW6     0x00a00000
#define QE_CR_SUBBLOCK_UCCSLOW7     0x00c00000
#define QE_CR_SUBBLOCK_UCCSLOW8     0x00e00000
#define QE_CR_PUSHSCH               0x0000000F
#define QE_CR_FLG                   0x00010000
#define READY_TO_CMD                0x00000000
#define QE_CR_SUBBLOCK_Rx           0x00020000

//Tx PushSched: Example for UCC1
IMM83XX->cp.cecdr = 0x80; //
IMM83XX->cp.cecr =QE_CR_SUBBLOCK_UCCSLOW1 | QE_CR_PUSHSCH |
QE_CR_FLG; // PushSched command for UCC Tx
while ((IMM83XX->cp.cecr & QE_CR_FLG) != READY_TO_CMD);
//Rx PushSched: Example for UCC1
IMM83XX->cp.cecdr = 0x82; //
IMM83XX->cp.cecr = QE_CR_SUBBLOCK_UCCSLOW1 | QE_CR_SUBBLOCK_Rx | QE_CR_PUSHSCH | QE_CR_FLG; //
PushSched command for UCC Rx
while ((IMM83XX->cp.cecr & QE_CR_FLG) != READY_TO_CMD);
```

Chapter 35
Inverse Multiplexing for ATM (IMA)

NOTE

ATM functionality is provided through an implementation of inverse multiplexing for ATM (IMA). This chapter provides a broad overview of the IMA specifications and the specific implementation.

35.1 Features

The IMA ATM Forum Specification defines the functions shown in Table 35-1.

Table 35-1. IMA Sublayer in Layer Reference Model

Layer	Sublayer	User Plane Functions	Layer Management Functions	Plane Management Functions
ATM	—	—	—	—
Physical	IMA Specific Transmission Convergence	• ATM cell stream splitting and reconstruction • ICP Cell Insertion & Removal • Cell Rate decoupling • IMA frame sync • Stuffing • Discarding bad-HEC cells	• IMA connectivity • ICP cell errors (OIF) • LIF/LODS/RDI-IMA defect processing • RDI-IMA alarm generation • Tx/Rx IMA link state report	• IMA group configuration • Link addition/removal • ATM cell rate change • IMA group failure notification • IMA statistics
	Interface Specific Transmission Convergence	• No cell discarding • No cell rate decoupling	—	—
		• Cell delineation • Cell scrambling & descrambling • Header error correction • HEC generation & verification	• HEC error indication • LCD-RDI alarm Generation	• LCD failure notification • TC statistics
	Physical Medium Dependent	• Bit timing • Line coding • Physical Medium	• Local alarm processing • RDI alarm generation	• Link failure notification • PMD state

This IMA microcode alone does not implement all of these functions. Software is responsible for managing the startup procedure, handling changes in group control, status, and maintaining the Link State Machine and Group State Machine. In general, this IMA microcode implements most of the IMA sublayer user plane functions and provides interrupts/statistics for the layer and plane management functions.

The key features of the IMA microcode are:

- Supports any UCC with multi-PHY UTOPIA capability
- Supports both IMA links and non-IMA links on the same UCC (on a per-PHY basis)

- Supports up to 8 IMA groups with one UCC
- Supports up to 31 links per IMA group using the internal microcoded TC layers (for an E1 configuration 2 IMA links can be supported per TDM interface)
- Supports up to 31 links per IMA group using an external TC layer device (Note that a maximum of 31 links can be supported per UCC)
- Performs the following IMA User Plane functions
  - ATM cell stream splitting and reconstruction
  - ICP cell insertion/removal – communication of control and framing information
  - Cell rate decoupling – insertion of ‘filler’ cells when ATM layer cells are unavailable
  - IMA frame synchronization – finding IMA frame boundaries within links
  - Stuffing--insertion of ‘stuff’ cells into fast links, in order to maintain an average data rate between links of a group
  - Discards cells with bad HECs
- Delay synchronization – correlating IMA frames among links of a group
- Maximum differential delay supported is user-programmable
- Provides interrupts on errors/state changes
- Maintains low-level statistic counters
  - Transmit stuff events
  - Receive stuff events
  - Receive ICP violations
  - Receive Out-of-IMA Frame anomalies

35.1.1 References

The features provided by the IMA microcode are driven by The ATM Forum’s IMA specification. The implementation of the IMA microcode is driven by the features, architecture, and resources available on the processor. In addition to published device errata, users should be familiar with the following (available at www.atmforum.com):

- The ATM Forum Technical Committee. Inverse Multiplexing for ATM (IMA) Specification Version 1.1 - AF-PHY-0086.001
- The ATM Forum Technical Committee. Inverse Multiplexing for ATM (IMA) Specification Version 1.0 - AF-PHY-0086.000

35.1.2 IMA Versions Supported

The IMA microcode supports IMA version 1.0 and version 1.1, with only minor configuration changes. These include the following:

- Programming the IMA version encoding in the OAM labels of the transmit ICP and Filler cell templates.
- Programming the validated OAM label field in the IMA group receive parameters.

35.1.3 PHY-Layer Devices Supported

The IMA microcode is primarily targeted at ATM's primary application (such as IMA over multiple DS1/E1). However, the IMA microcode supports any UTOPIA PHY device which (1) has a constant data rate and (2) can be programmed not to screen out HEC-errored cells. Most PHYs have this mode available for IMA also.

35.1.4 ATM Features Not Supported

The following ATM features are not available for IMA links only, but are available for non-IMA links:

- User-defined cells (UDC) (such as cells that are not 53 bytes and/or with customized headers)
- Internal rate mode for APC scheduling

35.2 IMA Protocol Overview

This section describes the IMA protocol, not this specific implementation of IMA microcode.

35.2.1 Introduction

Inverse multiplexing over ATM (IMA) is a cost-effective solution for carrying high speed connections such as T3/E3 and OC-3c/STM-1 links over already installed low speed connections such as T1/E1 links flexibly by dynamically adding/removing links depending on the bandwidth required. IMA transmits a stream of data over multiple low speed links and recombines the stream in the correct order at the end. IMA involves inverse multiplexing and de-multiplexing of ATM cells cyclically among links grouped to form a higher bandwidth logical link at a rate that approximates the sum of the link rates. Such groups of links are called IMA groups. Figure 35-1 illustrates the ATM inverse multiplexing technique in one direction. This technique applies in the opposite direction.

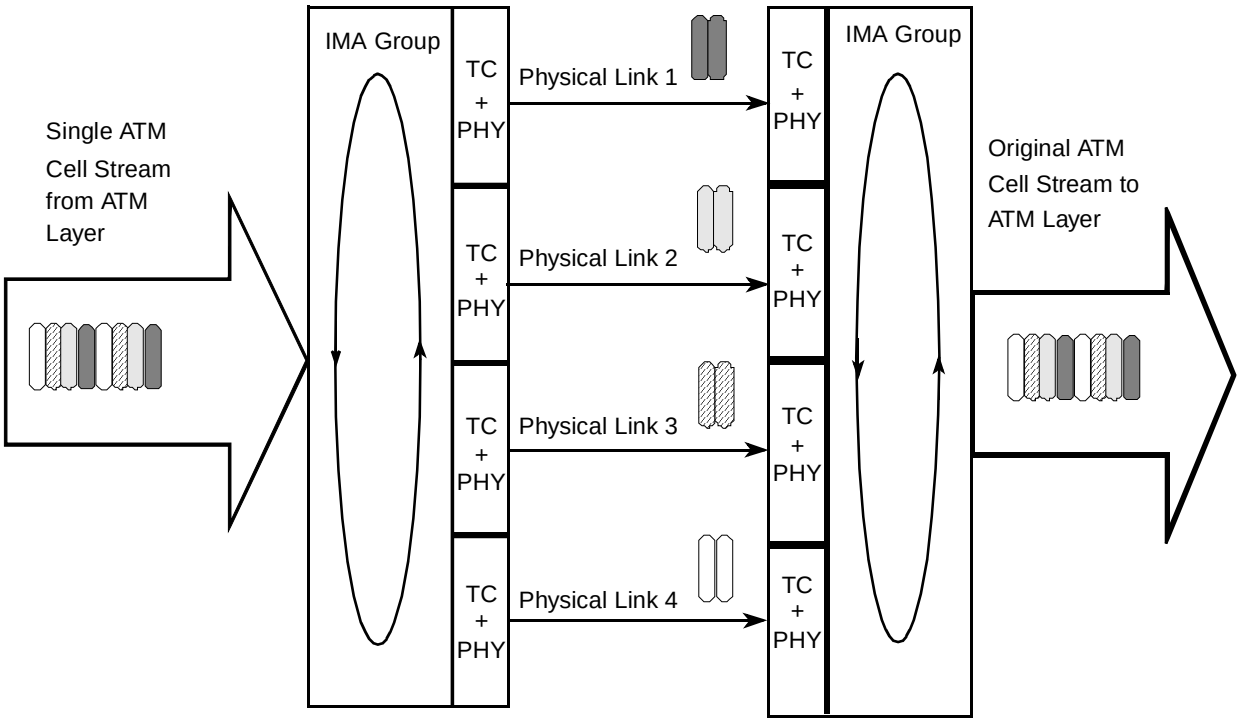


Figure 35-1. Basic Concept of IMA

In the transmit direction (near-end), the ATM cell stream received from the ATM layer is distributed, cell by cell, across the multiple links within the IMA group. At the far-end, the receiving IMA unit recombines the cells from each link, cell by cell, recreating the original ATM cell stream. The aggregate cell stream is then passed to the ATM layer. The IMA protocol enables the demultiplexing/deconstruction (transmit) of an ATM cell stream into multiple links. When receiving, the IMA protocol multiplexes/reconstructs incoming cells from multiple links into the original ATM cell stream. The IMA protocol must compensate for differences in clock rate and delay over the multiple links.

35.2.2 IMA Frame Overview

The IMA interface periodically transmits special cells that contain information to permit reconstruction of the ATM cell stream at the receiving end of the IMA virtual link. The receiver end reconstructs the ATM cell stream after accounting for the link differential delays, smoothing CDV introduced by the control cells, and so on. These cells, defined as IMA control protocol (ICP) cells, define an IMA frame. The transmitter must align the transmission of IMA frames on all links (shown in Figure 35-2) so that the receiver can adjust to differential link delays among the constituent physical links. Based on this required behavior, the receiver can detect the defensible delays by measuring the arrival times of the IMA frames on each link.

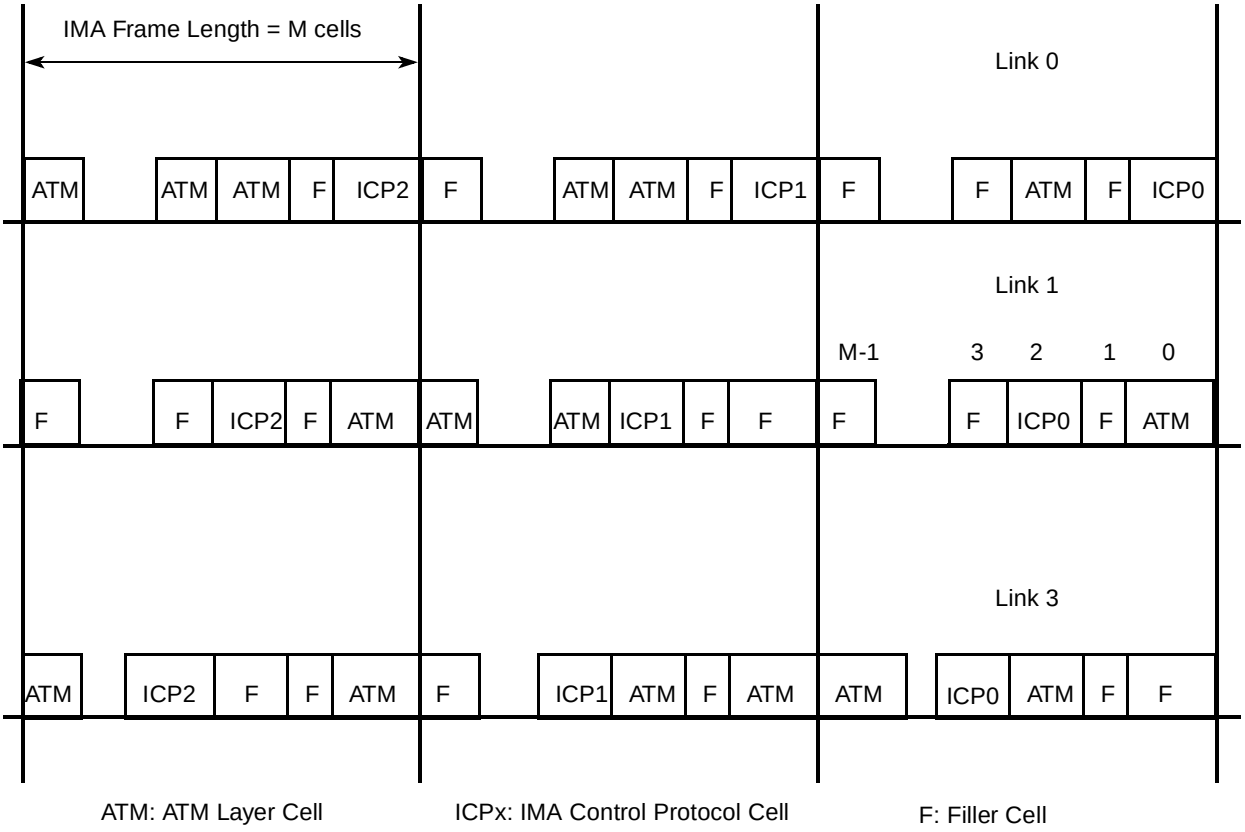


Figure 35-2. Illustration of IMA Frames

At the transmitting end, the cells are transmitted continuously. If there are no ATM layer cells to be sent between ICP cells within an IMA frame, the IMA transmitter sends filler cells to maintain a continuous stream of cells at the physical layer. The insertion of filler cells provides cell rate decoupling at the IMA sublayer. The IMA receiver should discard the filler cells. A new OAM cell is defined for use by the IMA protocol. The cell has codes that define it as an ICP or filler cell.

IMA data multiplexing is cell-based, where cells are distributed among the links in the IMA group in a round-robin cycle. To compensate for different clock rates, IMA must periodically insert ‘stuff’ cells into faster links to maintain a consistent average data rate over the links of the group. Furthermore, IMA must compensate for potential differences in delay between the links of the group. Per the IMA specification, the allowable delay differential for DS1/E1 links is 25ms, which at E1 rates is equivalent to approximately 118 cells. The IMA microcode allows the user to define the allowable delay differential through a delay compensation buffer of programmable length.

IMA accomplishes these goals by periodically inserting special OAM cells, which (among other things) define M-cell frame boundaries and provide frame sequence numbers and stuffing information. The receiver uses this framing information to correlate the received cell streams and extract cells in-order from the links of the IMA group, thereby reconstructing the original cell stream.

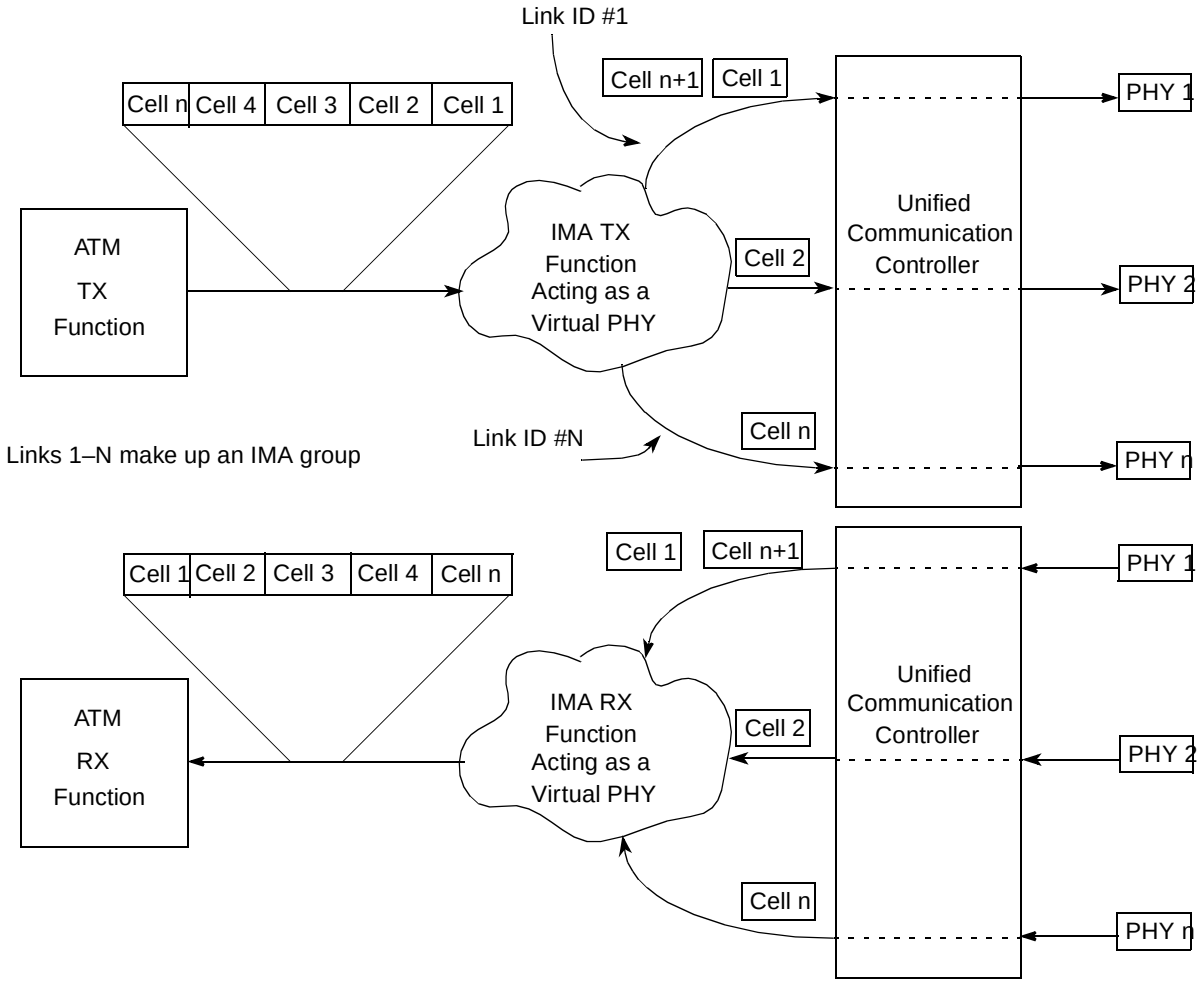


Figure 35-3. IMA Microcode Overview

35.2.3 Overview of IMA Cells

An IMA frame consists of M number of cells (M = 32, 64, 128, or 256 cells). Each frame consists of ATM data cells and IMA control cells. Two types of IMA control cells are defined and used by the IMA protocol: IMA control protocol (ICP) cells and filler cells.

35.2.3.1 IMA Control Cells

There is at least one IPC cell in each frame. An additional ICP cell may be included in a frame to compensate for timing differences between the links in an IMA group (for example, one link is slightly faster than the other). The insertion of additional ICP cells to compensate for timing differences between links is called a stuff event. The transmitter inserts stuff ICP (SICP) cells, and the receiver monitors for stuff indication and discards SICP cells. The location of the ICP cell in an IMA frame is determined during the IMA startup sequence.

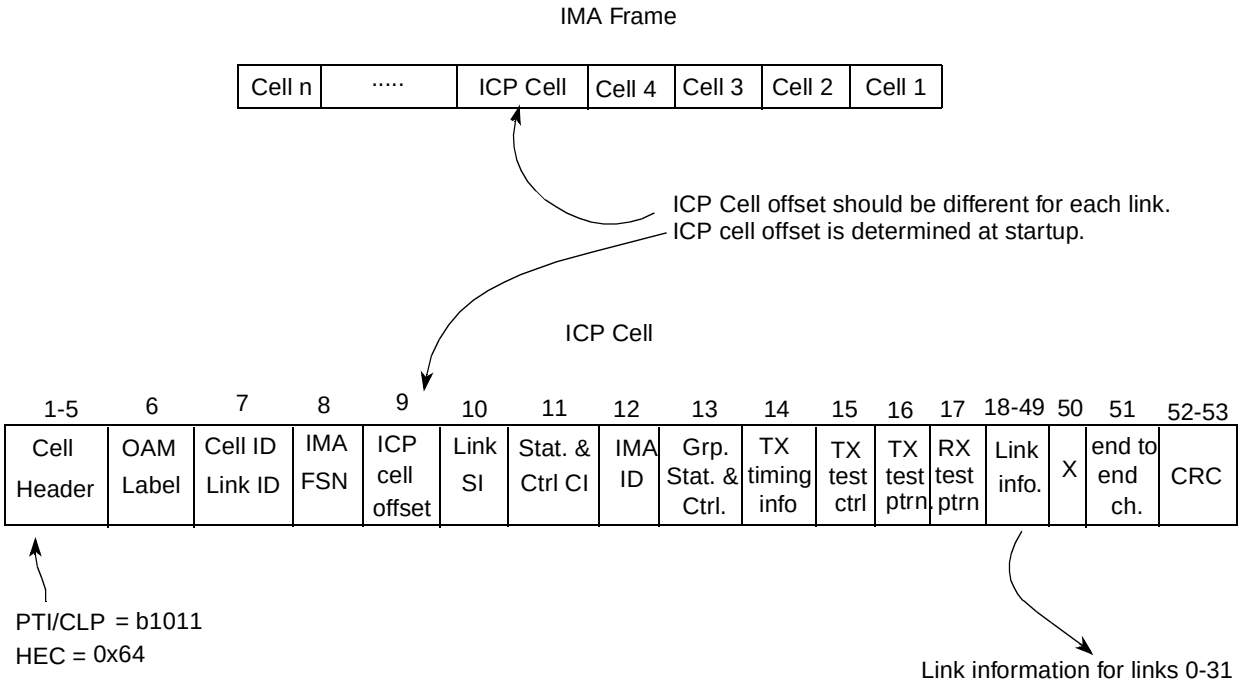


Figure 35-4. IMA Frame and ICP Cell Formats

The IMA protocol must compensate for potential differences in delay between the different links of the IMA group. The allowable delay differential for DS1/E1 links is 25ms, which at E1 rates is equivalent to approximately 118 cells.

35.2.3.2 IMA Filler Cells

At the transmitting end, cells are transmitted continuously. If there are no ATM layer cells to be sent between ICP cells within an IMA frame, then IMA transmitter sends filler cells to maintain a continuous stream of cells at the physical layer. The insertion of filler cells provides cell rate decoupling at the IMA sublayer. The IMA receiver should discard the filler cells.

35.3 IMA Microcode Architecture

This section explains the architecture of the receive and transmit IMA microcode tasks.

35.3.1 IMA Function Partitioning

The IMA microcode performs only functions with regular, critical real-time demands. The other functions of IMA, such as control and management, are the responsibility of host software. The IMA microcode corresponds primarily to the user plane functions defined in the IMA specification, and software must provide the layer management and plane management functionality. The IMA microcode provides interrupts when interaction with layer management and plane management is required.

35.3.1.1 User Plane Functions Performed by Microcode

- ATM cell stream splitting and reconstruction
- ICP cell insertion/removal
- Cell rate decoupling (that is, filler cell insertion/removal)
- IMA frame synchronization
- Stuffing
- Discards cells with bad HECs

35.3.1.2 Plane Management Functions Performed by Microcode

Although most plane management functions must be performed in software, certain statistics are intimately related to the lower-level user plane functions and are thus best provided by the microcode:

- ICP violations
- Transmit stuff events
- Receive stuff events

35.3.2 Transmit Architecture

This section discusses the behavior of the IMA microcode during transmission, focusing particularly on the independent transmit clock (ITC) mode of IMA. Differences in behavior when common transmit clock (CTC) mode is used are discussed at the end of this section.

Only one cell scheduler, known as the ATM pace controller or APC, is used per IMA group. This APC schedules transmission for the IMA group as a single aggregate channel. The APC hands these cells off to the IMA Tx microcode, which distributes these scheduled cells to each PHY in the IMA group. To compensate for clocking differences (jitter and average speed differential), the IMA Tx process distributes ATM cells into N jitter buffers with a depth of 5 cells. The IMA PHYs take cells from these jitter buffers and transmit them.

The cell scheduling is triggered by requests from the timing reference link (assertion of TxClav from the TRL PHY). Requests from non-TRL PHYs interact only with the jitter buffer and perform the stuffing function as needed (when the jitter buffer becomes too shallow). Therefore, the microcode tasks performed in response to TRL PHY requests and non-TRL PHY requests are different.

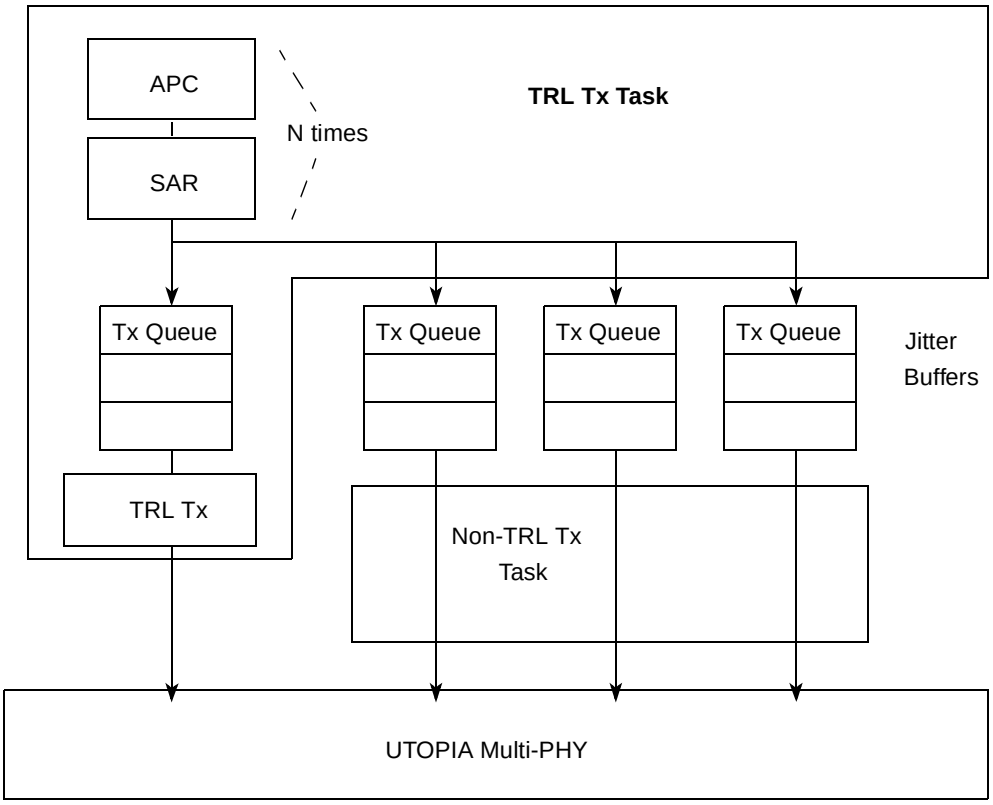


Figure 35-5. IMA Transmit Task Interaction

35.3.2.1 TRL Operation

A request from the TRL PHY triggers a complete round-robin process of cell scheduling and distribution, distributing one cell for each of the transmit queues of the N links in the IMA group. For each link, the microcode will:

1. Determine whether an ICP cell or a data/filler cell should be sent for the link:
   - If data/filler, determine whether link is in active or filler-only mode
   - If active, run the APC scheduling algorithm to find the next scheduled ATM channel
2. Distribute either:
   - An ICP cell
   - A filler cell (if ‘filler-only’ or ‘active’ with nothing scheduled in the APC)
   - A data cell (if a channel is scheduled in the APC, performing the appropriate segmentation task for the scheduled ATM channel, as determined by the channel’s AAL type)

The cells are distributed by writing the complete cells into circular transmit queues provided per link. These transmit queues function as jitter buffers to decouple the transmit rate of the TRL PHY from the transmit rate of the non-TRL PHYs in the group to allow for clocking differences between the PHYs.

At startup, the non-TRL links transmit filler cells until their transmit queues reach a minimum depth. To maintain less than the specified maximum +/-2.5 cell transmit timing differential (for cells within an IMA

frame), the TRL must exhibit the same behavior. Therefore, a 4-cell transmit buffer is also maintained for the timing reference link. The timing reference link begins to pass ATM layer cells to its PHY only after it has 3 cells in its buffer. Prior to this, it sends filler cells. This behavior is experienced only at group startup.

The TRL task also implements the standard amount of stuffing on the TRL link by maintaining a counter. When this task schedules (2048/ M) ICP cells for the TRL, a TRL stuff event is flagged and an indication of an upcoming stuff event is signaled in the ICP LSI field. If a TRL stuff event is flagged when the TRL task triggers, a stuff cell is sent to the TRL transmit queue. However, no cells are sent to the transmit queues of the non-TRL PHYs. This forces a standard amount of stuffing on the TRL, thereby reducing the effective data bit rate of the TRL to less than the minimum data clock rate allowed by the clock rate tolerance of the physical-layer standard. Therefore, this effective data bit rate is achievable by the non-TRL links; the non-TRL links can either stuff less if their data clock rate is slower than the TRL or stuff more if their data clock rate is faster than the TRL.

35.3.2.1.1 TRL Service Latency

NOTE

The functionality described in this section is available only with the latest RAM microcode package.

This optional feature allows the user to change the IMA APC behavior upon TRL request. When enabled the TRL request passes a programmable number of cells to the Tx queue of the links in an IMA group. This can be used to prevent the TRL from consuming a large amount of bandwidth before another cell is transmitted. The TRL request normally places a cell in a queue for N links where the group contains N links. Then a non_TRL link is free to pass a cell over the UTOPIA interface. The delay for the TRL can be long and in some cases the TC layer FIFO can underrun. This feature ensures that TRL and non-TRL requests are handled the same way, with 1 cell in and 1 cell out to the transmit queues. The non-TRL requests also trigger APC iterations when this feature is enabled. The depth of the TRL transmit queue must equal that of the non-TRL queues.

35.3.2.2 Non-TRL Operation

A request from a non-TRL PHY does not trigger any scheduling task. The cells for non-TRL links are already supplied (by the TRL task) in its associated transmit queue. The TRL simply read a cell out of its transmit queue and updates the queue pointers.

If the transmit queue becomes too shallow (because this link request rate is faster than the TRL), the link flags that a stuff event is imminent. The link signals an upcoming stuff event in the LSI field of its next ICP cell and then flags that a stuff event is due. The link continues sending cells from its queue as usual until it reaches its next ICP cell, upon which it indicates a stuff event in the ICP cell and transmits it but does not update the transmit queue pointers. When the link next requests a cell, the previous ICP cell is repeated (since the queue pointers were not updated). This process causes the transmit queue to deepen to the intended level.

At group startup, instead of accessing its transmit queue, the link sends filler cells to allow the transmit queues to reach their target steady-state depth. After the group startup flag is cleared, normal operation commences.

35.3.2.3 Transmit Queue Operation Examples (ITC mode)

The diagrams in this section demonstrate the different cases of queue operation and consequently justify the queue depth of 5 cells.

- The extraction pointer points to the queue entry currently supplied to the PHY. This cell must be entirely ready when the PHY requests it.
- The insertion pointer points to the queue entry to be filled next by the TRL process.

In the figures, note that the pointers and filled queue locations are just shown with respect to the overall queue depth available with the extraction pointer always shown at the bottom of the queue. This is done only for easy illustration. In reality, the transmit queues are circular queues in which the insertion and extraction pointers are continually rotating through the queue.

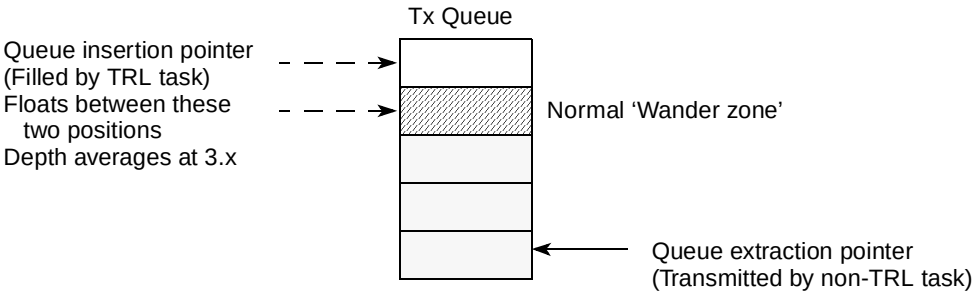


Figure 35-6. Transmit Queue Normal Operating State

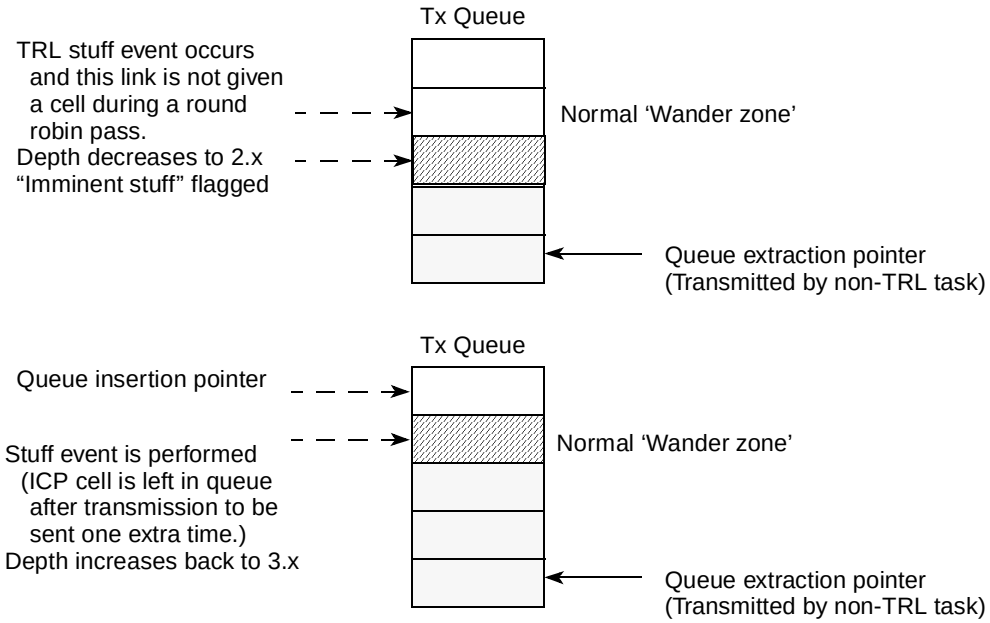


Figure 35-7. Transmit Queue Behavior: Link Clock Rate Same as TRL

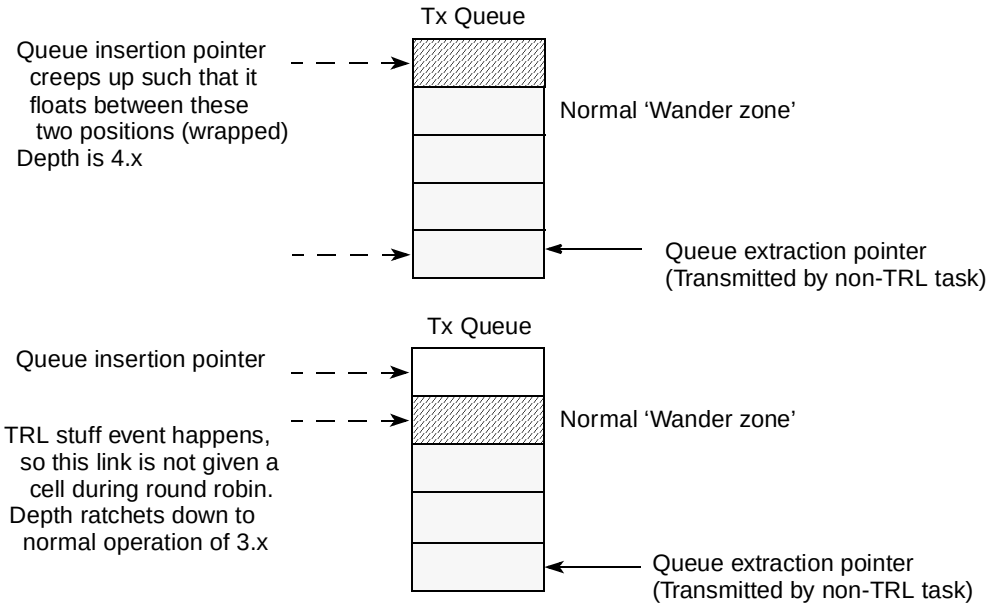


Figure 35-8. Transmit Queue Behavior: Link Clock Rate Slower than TRL

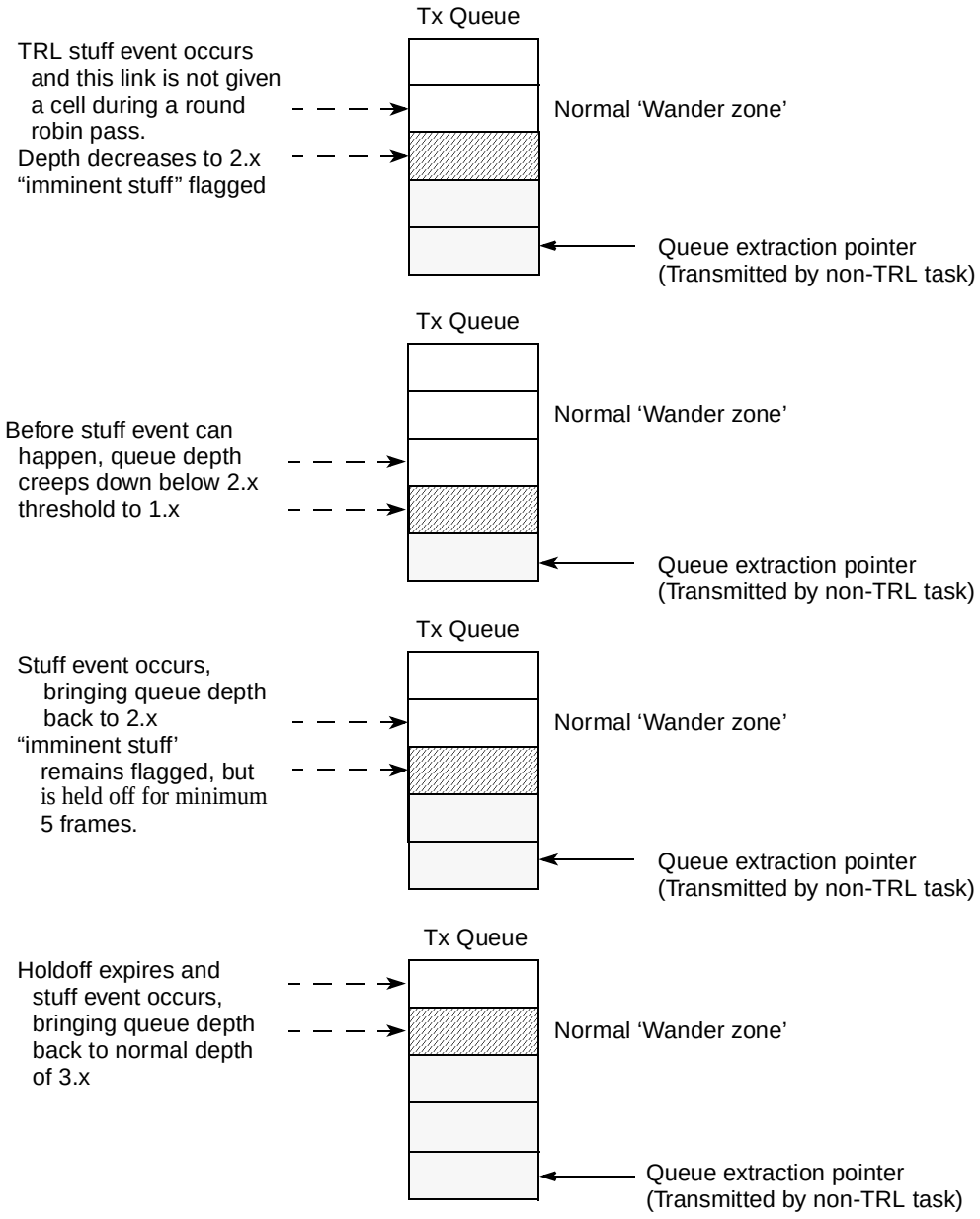


Figure 35-9. Transmit Queue Behavior: Link Clock Rate Faster Than TRL Worst-Case Event Sequence

35.3.2.4 Differences in CTC Operation

Overall, CTC operation is similar to ITC operation in task partitioning and structure. Differences are as follows:

- Transmit buffers are still maintained for the non-TRL links, but these buffers do not jitter because the transmit clocks are all the same. However, queue depths may vary slightly because PHY requests, while synchronous, do not necessarily come in simultaneously (may have constant offsets) and are definitely not serviced simultaneously.

- The non-TRL tasks do not determine when to perform stuffing on the non-TRL links. When the TRL flags imminent stuff on its own link, it flags imminent stuff on all the non-TRL links as well. Thus, the non-TRL links also stuff their links every 2048 cells.
- The non-TRL queue depths are the same as the TRL queue (4 cells).

35.3.3 Receive Architecture

The receive task consists of three parts:

- Cell reception from the link.
- Trigger for activating the cell processing task, including timing recovery if desired.
- Actual cell processing (passing cells to the ATM layer).

The cell reception task services requests from the links (through the UTOPIA multi-PHY interface and the UCC), maintains the link state, and (if the link and group state dictates) writes the received cells into the link delay compensation buffer in external memory. The cell processing activation function coordinates passing cells from the cell reception task on an on-demand basis. The cell processing task extracts cells from the delay compensation buffers and passes them to the ATM layer for processing.

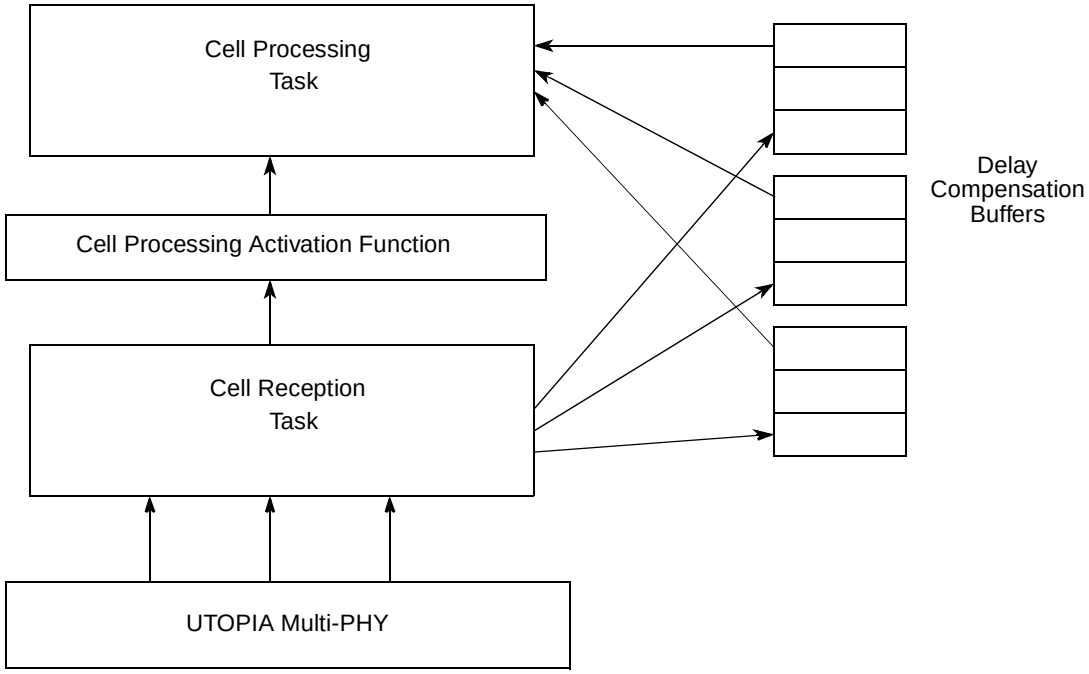


Figure 35-10. IMA Receive Task Interaction

35.3.3.1 Cell Reception Task

In the cell reception task, received ICP cells in which an SCCI change is noted, except for the first ICP received cell, are passed to a user-defined receive AAL0 channel to be processed by software. These received cells (and other event indications) are used by the software to initialize IMA links and groups, and to manage transitions between link and group states.

The cell reception task centers around a four-state link state machine. Microcode tasks are performed within each state, but transitions between states are managed by software. The processing of received cells (both ICP cells and data cells) is determined by the state of the link.

NOTES:
— Each IMA link follows a four-state machine.
— The driver processes each received ICP cell (AAL0-specific channel) to control the state machine.
— According to the link state, the microcode executes different tasks.

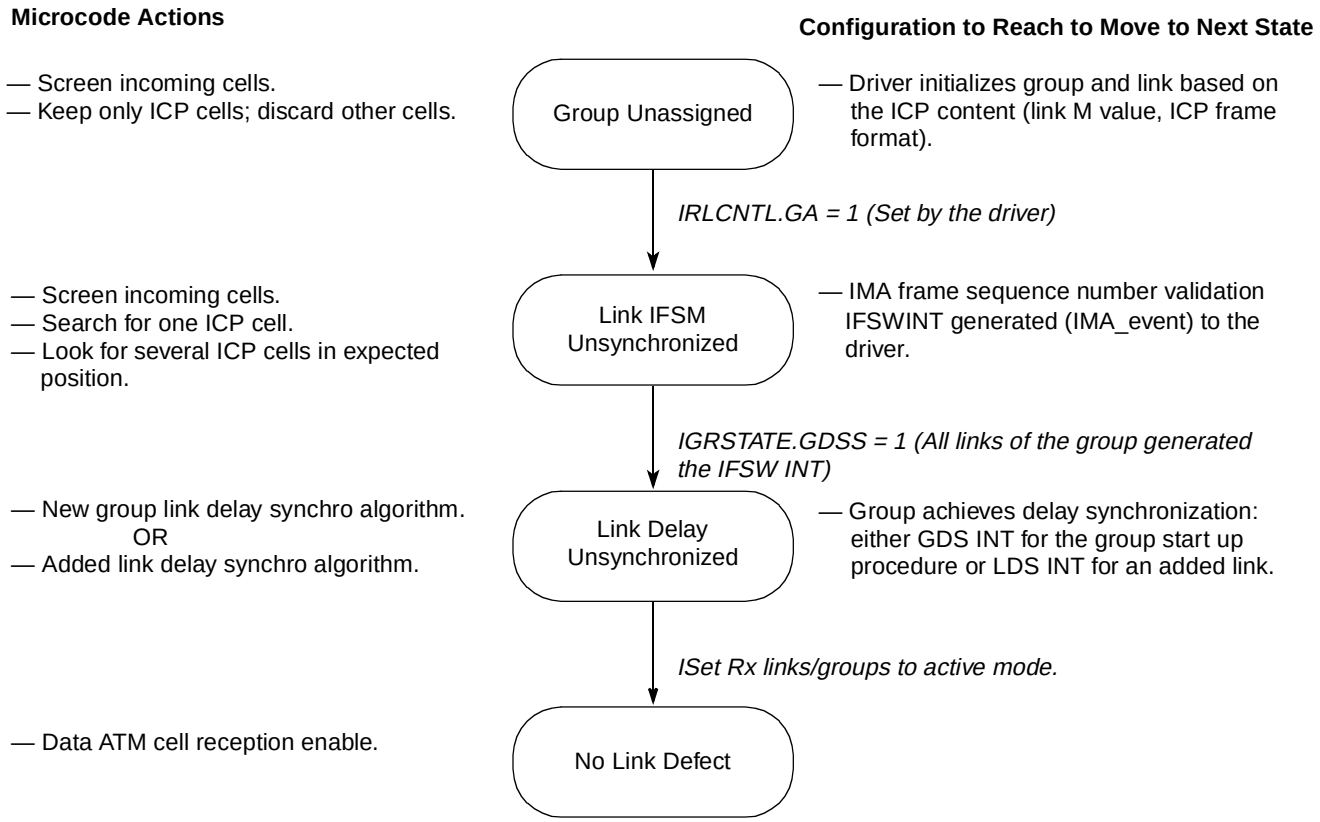


Figure 35-11. IMA Microcode: Receive Process

As Figure 35-11 shows, the states are as follows:

- Group Unassigned—Corresponds to a link that is known to be IMA but for which no information is known, such as IMA group number and IMA frame size. The receive task only screens the incoming cells for ICP cells and discards all others. ICP cells in which an SCCI change is noted are passed to a user-defined receive channel, where software must interpret their content to initialize the link and add it to the group. The link transitions to the ‘Link IFSM Unsynchronized’ state when software sets its Group Assigned flag.
- Link IFSM Unsynchronized—The link state machine has not yet found or validated the frame boundary for this link. Software initially enters this state after it defines the link M value and expected ICP cell format and then sets the Group Assigned flag. This state can be subsequently re-entered as the result of a link defect. The link searches for an ICP cell, then looks for ICP cells in the expected position of subsequent frames to validate frame synchronization. Cells other than ICP cells are discarded. When the IFSM becomes synchronized, an interrupt to the host is

generated. The link transitions to the Link Delay Unsynchronized state when software appropriately sets its group and link synchronization flags.

- Link Delay Unsynchronized—Contains two separate operations, depending on the state of the group to which the link belongs. If the link enters this state as the result of group startup, it performs the new-group link delay synchronization algorithm. If the link enters this state while the group is already activated per the link addition/slow recovery (LASR) procedure of the IMA standard, it performs the added-link delay synchronization algorithm. As the result of either algorithm, an interrupt is generated to the host when delay synchronization is achieved.
- No Link Defect—Normal receive operation after the link is established and the link-state is appropriately communicated to the far end. If the link is not inhibited at the group or link level, reception of ATM cells occurs. If the link or group is inhibited, non-ICP received cells are replaced with filler cells. Cells received (or their replacements) are written to the link delay compensation buffer. The link transitions out of this state only as the result of an error or a reset by host software.

35.3.3.2 Cell Processing Activation Function

The cell processing task is triggered directly by the cell reception task if a cell is written to a delay compensation buffer. Per the IMA specification, this is allowable under both of the following conditions:

- The IMA receiver is directly built into end equipment that directly terminates the ATM layer (that is, terminates all ATM connections).
- The system can carry only services that either do not require CDV control (some data services) or CDV control is handled in some other way, such as absorbed into a play-out buffer at the ATM layer connection termination.

This processor may qualify as such a system if it terminates all ATM connections that it receives. The buffer descriptors and external memory serve as a play-out buffer. Furthermore, a system that does not terminate cells but instead passes cells port-to-port can also use this mode of operation if either of the following conditions is met:

- All cell streams are switched at the VC level only, and the VC traffic type is supported by this APC. The APC can be programmed to reshape the VC appropriately at the egress port. Therefore, no cell delay variation (CDV) is introduced.
- Some (or all) cell streams are switched at the VP level, but the switched VPs carry only traffic for which cell delay variation (CDV) within the bounds of an IMA round-robin distribution is tolerable. For example, if the IMA group consists of 8 DS1 links, the maximum CDV introduced by this method would be 8 cell times, or approximately 2.2 ms

If the system meets these qualifications, this mode of operation is recommended because it is the simplest and yields overall better system performance. That is, this mode requires less QUICC Engine processing power.

35.3.3.3 Cell Processing Task

The cell processing activation function triggers the cell processing task. This task extracts cells in-order from the delay compensation buffers. These cells are processed per the processor standard ATM operation; that is, they are mapped into ATM channels and processed per the appropriate AAL or OAM function. If

the on-demand cell processing activation function is used and the cell processing task is triggered, it extracts cells in-order from the delay compensation buffers until either no more are available or four cells are extracted. The purpose of limiting the cell processing task to a maximum of four cells is to limit the maximum latency of servicing requests from the external PHYs. On the average, the receive process delivers one cell to the ATM layer per cell reception.

35.4 IMA Programming Model

35.4.1 Data Structure Organization

Figure 35-12 shows the organization of IMA data structures.

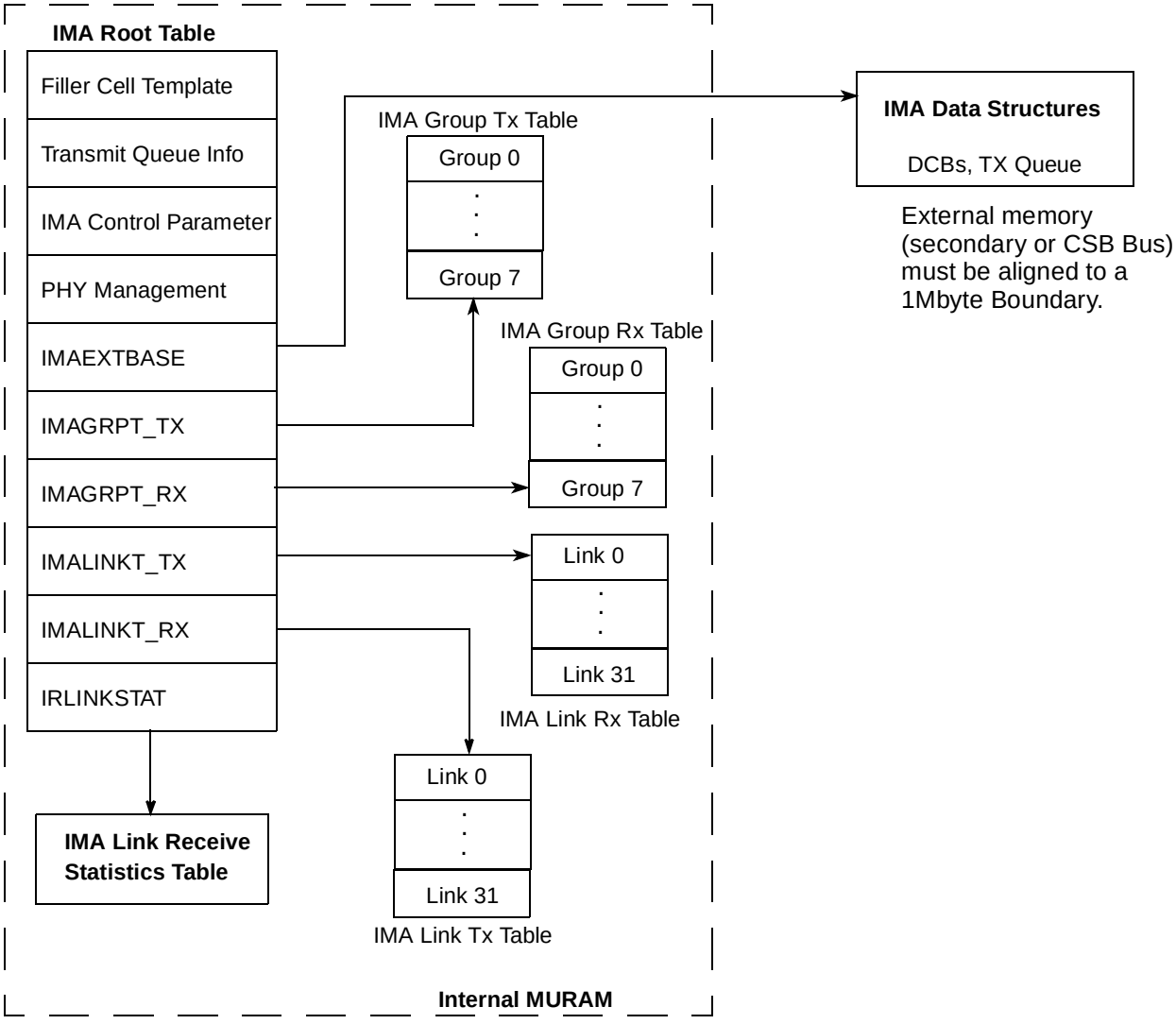


Figure 35-12. IMA Root Table Data Structures

The IMA data structures are organized as follows:

- Parameters at the UCC level determine common ATM parameters and location of the IMA root table.
- IMA root table includes parameters used by all IMA links of this UCC.
- IMA group receive table and IMA group transmit table entries include parameters that define the receive and transmit states and settings of the associated IMA group.
- IMA link receive table and IMA link transmit table entries include parameters that define the receive and transmit states and settings of the associated IMA link

35.4.2 IMA UCC Programming

35.4.2.1 UCC Registers

The UCC protocol-specific mode register (UPSMR) for ATM operation is described in Section 31.6.5.1, “UPDCx in ATM Protocol.” Refer to that section for information pertaining to IMA.

35.4.2.2 UCC Parameters

User software must also configure the IMA_Temp parameter in the UCC distributor local parameter table. IMA functionality is enabled through the GMODE register. Refer to Section 30.3.2.5, “Global Mode Entry (GMODE).”

35.4.2.3 IMA-Specific UCC Parameters

The following parameter must be programmed in the UCC parameter RAM page in addition to the standard UCC parameters for ATM.

Table 35-2. UCC Parameter RAM Additions

Offset	Name	Width	Description
0x04	IMAROOT	Hword	Offset of IMA root table in multi-user RAM. The IMAROOT is offset from the UCC Sub-page0 Configuration Table. Must be 128-byte aligned.

35.4.2.4 Differences From CPM-Based IMA Implementation

The following text lists the differences of the IMA implementation for the QUICC Engine block versus the CPM:

- No alignment constraint on IMAROOT ending with 0x80; the IMA root can reside on any 128 byte aligned address.
- TXPHYEN and RXPHYEN bit arrays are removed from the IMA root.
- IMAPHY parameter no longer exists because the UTOPIA address is compressed to the range 0–31 per UCC using tables defined for Rx and Tx.
- Added the serial ATM enable bit to the IMACTL.
- No support for IDCR mode.

35.4.3 IMA Root Table

Table 35-3. IMA Root Table[1]

Offset	Name	Width	Description
-0x04	**IMAFILLERHD**[2]	4 Bytes	Filler cell template. Used by microcode in transmission of filler cells. The cell is formatted as byte-swapped, and additionally the header is bitswapped. [This is due to hardware implementation, and does not imply the order of the transmission of the cell. The transmission of the cell is per the ATM standard.] Content should be: 0xD0000000 (header) 0x6A6A0001 or 0x6A6A0003 (for IMA Version 1.0 or 1.1) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0xC6026A6A or 0xD9026A6A (for IMA Version 1.0 or 1.1)
0x00–0x2F	IMAFILLERPLD	48 Bytes	
0x30	FILLTAG	Byte	Tag indicating that the filler template is a filler cell. Program to zero.
0x31	**TQ_SIZE**	Byte	Transmit queue size. Recommended value is 0x18. Must be a multiple of 4. Refer to Section 35.4.6.1, “Transmit Queues” for more details
0x32	**TQ_TARGET**	Byte	Transmit queue target level. Recommended value is 0x0C. Must be a multiple of 4.
0x33	**TQ_THRESHOLD**	Byte	Transmit queue stuff threshold. Recommended value is 0x0C. Must be a multiple of 4.
0x34	RESERVED	Hword	Reserved.
0x36	**IMACNTL**	Byte	IMA control parameter. Controls functions shared by all IMA groups for this UCC.
0x37	TMP_PCNT	Byte	Microcode managed parameter (pass count).
0x38	**RXPHYEN_TSIZE**	Byte	Receive PHY Enable Table Size. Number of byte entries programmed in the receive table for IMA PHY number selection for IMA links on this UCCs UTOPIA bus. If the table consist of 1 entry program this parameter to 0, if it contains 2 entries program to 1 etc. See Section 35.4.3.2, “Utopia PHY Address Compression” for more details.
0x39	**RXPHYEN_TPTR**	3 Bytes	Receive PHY Enable Table Pointer. Pointer to table in the MURAM containing UTOPIA PHY addresses for the purpose of compression to the range 0-30. See Section 35.4.3.2, “Utopia PHY Address Compression” for more details.
0x3C	**TXPHYEN_TSIZE**	Byte	Transmit PHY Enable Table Size. Number of byte entries programmed in the transmit table for IMA PHY number selection for IMA links on this UCCs UTOPIA bus. If the table consist of 1 entry program this parameter to 0, if it contains 2 entries program to 1 etc. See Section 35.4.3.2, “Utopia PHY Address Compression” for more details.
0x3D	**TXPHYEN_TPTR**	3 Bytes	Transmit PHY Enable Table Pointer. Pointer to table in the MURAM containing UTOPIA PHY addresses for the purpose of compression to the range 0-30. See Section 35.4.3.2, “Utopia PHY Address Compression” for more details.
0x40–0x43	—	—	Reserved. Must be programmed to zero during initialization.

Table 35-3. IMA Root Table[1] (continued)

Offset	Name	Width	Description
0x44	**IMAEXTBASE**	Word	IMA external structure base pointer. Points to region in external memory where external IMA data structures are located. Must be aligned to a 1MB boundary (such as program bits 12-31 to zero).
0x48	**IMAGRPT_TX**	Hword	Offset of IMA group transmit table in multi-user RAM (MURAM). Must be 16-byte aligned.
0x4A	**IMAGRPT_RX**	Hword	Offset of IMA group receive table in MURAM. Must be 64-byte aligned.
0x4C	**IMALINKT_TX**	Hword	Offset of IMA link transmit table in MURAM. Must be 32-byte aligned.
0x4E	**IMALINKT_RX**	Hword	Offset of IMA link receive table in MURAM. Must be 32-byte aligned.
0x50	**IRLINKSTAT**	Hword	Offset of the optional IMA link receive statistics table in MURAM. Must be 8-byte aligned.
0x52	TMP_LPTR_RX	Hword	Microcode-managed parameter. Temporary storage of link table pointer.
0x54	TMP_LPTR_TX	Hword	Microcode-managed parameter. Temporary transmit table pointer.
0x56	TMP_GPTR_TX	Hword	Microcode-managed parameter. Temporary transmit group pointer.
0x58	TMP_GPORD_TX	Hword	Microcode-managed parameter. Temporary transmit group order pointer.
0x5A	TMP_GPTR_RX	Hword	Microcode-managed parameter. Temporary receive group pointer.
0X5C	TMP_RTRN_RX	Hword	Microcode-managed parameter. Temporary return pointer.
0x5E	TMP_GPTR2_RX	Hword	Microcode-managed parameter. Temporary receive group pointer 2.
0x60–0x6B	—	—	Reserved. Must be programmed to zero during initialization.
0x6C	**ITPGRPO**	Hword	Required for optional TRL Service Latency enhancement only. IMA Temp Group Order - Points to the base of a 2byte temp pointer storage per group. Software initialized before UCC is enabled. Microcode managed parameter.
0x6E–0x7F	—	—	Reserved. Must be programmed to zero during initialization.

1 **Boldfaced** entries in this table indicate parameters that must be initialized by the user. All other parameters are managed by the microcode and should be initialized to zero unless otherwise stated.

2 IMAFILLERHD is located at the word immediately preceding the 128-byte aligned region defined by IMAROOT. Thus it is located at offset 0x04 from the base o the f IMAROOT table.

35.4.3.1 IMA Control (IMACNTL)

The fields of the IMACNTL are shown in Figure 35-13.

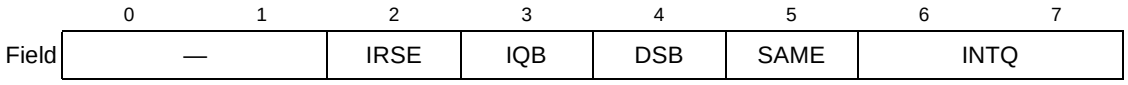


Figure 35-13. IMA Control (IMACNTL)

Table 35-4 describes the IMACNTL bit fields.

Table 35-4. IMACNTL Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	IRSE	IMA link receive statistics enable. If link receive statistics gathering is disabled, there is no need to initialize IRLINKSTAT or reserve space for the receive statistics table. 0 Link receive statistics gathering is disabled. 1 Link receive statistics gathering is enabled.
3	IQB	Interrupt queue bus. Defines the bus on which the IMA interrupt queue is located. 0 On the CSB bus. 1 On the secondary bus.
4	DSB	Data structure bus. Defines the bus on which the IMA external structure memory area is located. 0 On the CSB bus. 1 On the secondary bus.
5	SAME	Serial ATM Enable. 0 The TC layer is not SAM based 1 The TC layer is SAM based, that is, an MTC.
6–7	INTQ	Number of the ATM interrupt queue dedicated to IMA events. Note that these events do not include ICP cell reception events; the handling of ICP cell reception events is programmed in the RCT of the ICP channel defined by RICPCH.

35.4.3.2 Utopia PHY Address Compression

In IMA mode, the maximum number of IMA links supported per UCC is 31. The PHY address used by the IMA links must be in the range 0-30. As the UTOPIA address for a PHY is in the range 0-127 the IMA PHY address therefore must be compressed to the range 0-30. Figure 35-14 depicts the compression scheme.

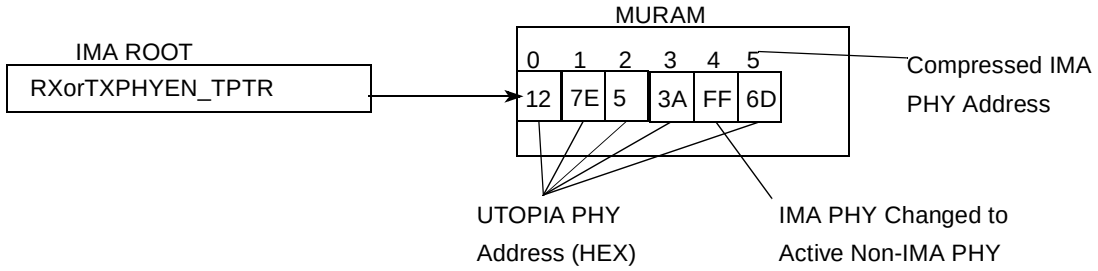


Figure 35-14. Utopia PHY Address Compression Tables

Each entry in the UTOPIA address compression table is 1 byte. The contents of a table entry is the UTOPIA PHY address for links that are IMA based on this UCC. The byte offset of a matched UTOPIA address from the table base (defined by RXPHYEN_TPTR or TXPHYEN_TPTR) is the IMA PHY address. For example in Figure 35-14 UTOPIA PHY address 0x7E is mapped to IMA link/PHY 0x1. Therefore, PHY address 0x7E uses link table 1 to define its IMA structures. The compressed address values must be used in the transmit and receive group order tables and not the UTOPIA PHY address.

There are two tables, one each for transmit and receive, but if the same UTOPIA PHY addresses are used on transmit and receive for IMA links, one compression table can be used, with RXPHYEN_TPTR and TXPHYEN_TPTR programmed to the same value.

If a link in an IMA group is then removed and therefore disabled at the TC level, the compression table must be updated before the link can become active again as a non-IMA PHY. Before re-enablement, the old link offset in the compression table must be programmed to a value that does not yield a match in the table. The suggested value is 0xFF. The link cannot be removed from the compression table by resizing because other links may still be in the IMA group and their offset within the table cannot be changed. The offset cannot be changed because it defines the link IMA PHY number, which is subsequently used to access the corresponding link table. If the link is added again as an IMA PHY, no compression table updates are required.

When links are added to an existing group, the table can be resized on the fly by first adding the new UTOPIA PHY address to the end of the compression table and then changing the xPHEN_TSIZE to reflect the new table size. Any already existing PHYs in the compression table must not have their relative offsets from the base of the compression table changed. Any added PHY must be placed either at the first entry at the end of the table or an entry beyond the first free entry. For example, consider an IMA group consisting of two links mapped as follows:

IMA PHY 0 = UTOPIA PHY ADDRESS 0x66

IMA PHY 1 = UTOPIA PHY ADDRESS 0x5A

If another PHY is to be added to the group (for example, 0x3F) and this PHY is required to be IMA PHY 3, the following table should be constructed:

0x66 | 0x5A | 0xFF | 0x3F

In the example, the xPHEN_TSIZE should be programmed to 3 (after the compression table is updated) to reflect a table size of four entries. The unused entry corresponding to IMA PHY 2 should be programmed to a value of 0xFF to pad the table so the UTOPIA PHY 0x3F is IMA PHY 3 after compression. At a later point, another link that uses IMA PHY 2 can be added to the group. In this case after the offset for IMA PHY 2 is overwritten in the compression table, there is no need to update the corresponding xPHEN_TSIZE parameter.

35.4.4 IMA Group Tables

The IMA group tables consist of multiple IMA group structures indexed by group number, which ranges from 0–7. The transmit and receive parameters are located in separate tables. The IMA group transmit table entries are 16 bytes long. The IMA group receive table entries are 64 bytes long. However, there is no need to reserve memory space in MURAM for 8 receive IMA groups if less than 8 IMA groups are required. To conserve memory space used by these tables, it is best to add groups starting from group zero. Note that group number is independent of the assignment of IMA ID.

35.4.4.1 IMA Group Transmit Table Entry

Table 35-5. IMA Group Transmit Table Entry [1]

Offset	Name	Width	Description
0x00	**IGTCNTL**	Byte	IMA group transmit control parameters.
0x01	IGTSTATE	Byte	IMA group transmit state. Microcode-managed parameter. Must be initialized to zero at group startup.
0x02	**TGRPORDER**	Hword	Offset of transmit group order table in MURAM. Can be changed on the fly.
0x04	**TVPHYNUM**	Byte	Transmit Virtual PHY number. Maps this IMA transmit group to a virtual PHY number for the purpose of selecting an ATM pace controller (APC) table number for this IMA group. Note that this parameter must be unique; it must not conflict with the PHY number of any non-IMA PHYs or with the TVPHYNUM of any other IMA group. If there are any PHY numbers which will never be used in a system (such as the actual PHY with the address does not exist), then TVPHYNUM should be selected from one of these PHYs. If no such PHY numbers are available (such as the multi-PHY interface consists of the full 31 PHYs), then select TVPHYNUM from one of the PHY numbers of the PHYs within this IMA group.
0x05	TIFSN	Byte	Transmit IMA Frame Sequence Number (IFSN). Microcode-managed parameter. Increments each time an IMA frame is transmitted, cycling from 0 through 255. Should be initialized to zero at group startup.
0x06	TMCTR	Byte	Transmit IMA M counter. Microcode-managed parameter. Tracks IMA frame boundaries. Increments once per round-robin distribution of cells to the transmit queues, cycling from 0 through M. Initialize to zero at group startup.
0x07	TRLSTFCNT	Byte	TRL stuff frame counter. Microcode-managed parameter. Controls required stuffing on the TRL. Decrements each time an ICP cell is sent on the TRL, cycling from TRLSTFN through 0. A TRL stuff event occurs when it reaches zero. Initialize to the value of TRLSTFN at group startup.
0x08	**TICPPTR**	Hword	Offset of transmit ICP cell payload template area in MURAM. Must be 64-byte aligned. This parameter and the associated template area may only be changed when IGCNTL[ICPC]=IGTSTATE[ICPCA]. After changing TICPPTR, IGCNTL[ICPC] must be toggled.
0x0A	**TM**	Byte	Transmit IMA frame size. Program to 31, 63, 127, or 255 for frame sizes (M) of 32, 64, 128, or 256, respectively.
0x0B	**TRLSTFN**	Byte	TRL stuff frame number. Defines the number of IMA frames sent between TRL stuff events. For ITC operation IGTCNT[CTC] = 0, program TRLSTFN = 2048/M. For CTC operation IGTCNT[CTC] = 1, program TRLSTFN = (2048/M) – 1. Refer to Section 35.4.4.1.1, “IMA Group Transmit Control (IGTCNTL).”
0x0C	**TNUMLINKS**	Byte	Number of transmit links in the IMA group that are in the active state (ILTCNTL[TXSC]=01). Used by the APC to scale the rescheduling parameters appropriately when rescheduling channels.
0x0D	RTSTPCNT	Byte	Real time-stamp subcounter. Microcode-managed parameter, used by the APC. Initialize to zero at group startup.

Table 35-5. IMA Group Transmit Table Entry (continued)[1]

Offset	Name	Width	Description
0x0E	**IASNCCtr**	Byte	Required for optional TRL Service Latency enhancement only. IMA APC Scheduled Number of Cells Counter - Number of cells passed to the groups links upon request. Initialize to IASNC.
0x0F	**IASNC**	Byte	Required for optional TRL Service Latency enhancement only. IMA APC Scheduled Number of Cells - reset for IASNCtr. Number of cells passed to the groups links upon request. Recommended value is 1.

1 **Boldfaced** entries must be initialized by the user. All other parameters initialize to zero.

35.4.4.1.1 IMA Group Transmit Control (IGTCNTL)

The fields of the IGTCNTL register are shown in Figure 35-15.

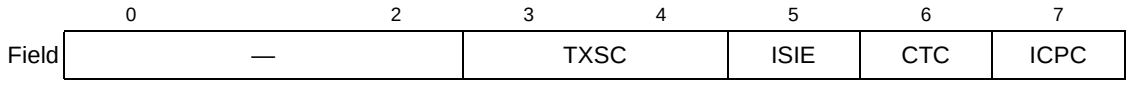


Figure 35-15. IMA Group Transmit Control (IGTCNTL)

Table 35-6 describes the IGTCNTL bit fields.

Table 35-6. IGTCNTL Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–4	TXSC	Transmit status/control. Sets the transmit mode of the IMA group. 00 Filler mode. The IMA group transmits only ICP and filler cells, no data cells. 01 Active mode. The IMA group is capable of sending data cells. 1X Reserved.
5	ISIE	IMA Scheduler Split Iterations Enable 0 APC is not split, TRL completes round robin distribution of cells. 1 APC split, both TRL and non-TRL requests distribute cells to the transmit queues. Required for optional TRL Service Latency. This option is valid for a 1 IMA Tx group configuration per UCC. If more than 1 IMA group is used per UCC then clear ISIE.
6	CTC[1]	Transmit clock mode for this IMA group. 0 Independent transmit clock (ITC) mode. 1 Common transmit clock (CTC) mode.
7	ICPC	ICP change flag. Maintains at least the minimum 2-frame spacing of ICP cell control/status changes. Initialize to zero at group startup. Must be toggled by software whenever TICPPTR is changed. After two subsequent IMA frames are transmitted, the IGTSTATE[ICPCA] field will be changed to match IGTCNTL[ICPC]. When IGTCNTL[ICPC] equals IGTSTATE[ICPCA], changes to TICPPTR are allowed.

[1]Ensure transmit clock mode is not changed during IMA group startup to avoid erratic behavior.

35.4.4.1.2 IMA Group Transmit State (IGTSTATE)

The fields of the IGTSTATE register are shown in Figure 35-16.

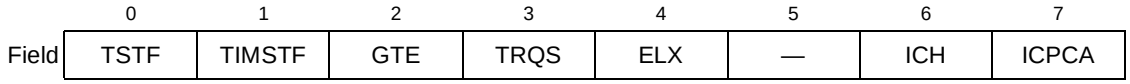


Figure 35-16. IMA Group Transmit State (IGTSTATE)

Table 35-7 describes the IGTSTATE bit fields.

Table 35-7. IGTSTATE Field Descriptions

Bits	Name	Description
0	TSTF	TRL stuff flag. Microcode-managed parameter. Indicates that the next ICP cell on the TRL will be part of a stuff event. Initialize to zero at group startup.
1	TIMSTF	TRL imminent stuff flag. Microcode-managed parameter. Indicates that an upcoming stuff event will be signaled in the next ICP of the TRL (via LSI=001). Initialize to zero at group startup.
2	GTE	Go to end flag - TRL has requested 2 time before round robin distribution has completed or link will underrun and is still due cell from round robin distribution. Microcode managed parameter initialize to 0. Used for optional TRL Service Latency enhancement only.
3	TRQS	TRL Request - TRL has requested therefore 1 round robin distribution of cells is yet to be completed. Microcode managed parameter initialize to 0. Used for optional TRL Service Latency enhancement only.
4	ELX	Early exit flag. Microcode-managed parameter. Initialize to zero at group startup.
5	—	Reserved
6	ICH	ICP change holdoff. Microcode-managed parameter. Initialize to zero at group startup.
7	ICPCA	ICP change allowed flag. Microcode-managed parameter. See description of IGTCNTL[ICPC].

35.4.4.1.3 Transmit Group Order Table

The transmit group order table defines the order of the links in the round-robin distribution of cells to the links of the IMA group. The table consists of an array of bytes ordered from first link to last link, with each byte providing the PHY address of its associated link. The end of the table is indicated by an entry programmed to 0x1F.

This table alone defines the order of cell distribution. It is the responsibility of software to program the LID of the links in the IMA Link Table entries, and to program the group order table such that its order corresponds to the order of increasing LIDs within the group.

The format of a transmit group order table entry is shown in Figure 35-17.

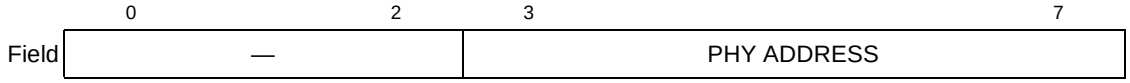


Figure 35-17. Transmit Group Order Table Entry

Table 35-8 describes the format of a transmit group order table entry.

Table 35-8. Transmit Group Order Table Entry Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	PHY ADDRESS	PHY address (Up to 31 PHYs[0–30]) of the link transmitting in this position in the round-robin. A value of 0x1F in this field indicates the end of the group order table. The value programmed must be the compressed value after Utopia Address Compression see Section 35.4.3.2, “Utopia PHY Address Compression.”

35.4.4.1.4 ICP Cell Templates

The ICP cell templates are areas of memory provided by software to the microcode for construction of ICP cells for transmission. Software prepares the fields which are common to the group (such as class B, C, D, and E parameters). The other (class A) parameters will be written to the appropriate fields by the microcode. The template is 64 bytes long and must be aligned on a 64-byte boundary.

Table 35-9 describes the format of a group order table entry. The ICP cell template is formatted as byte-swapped, and additionally the ICP cell header is bit swapped. [This is due to hardware implementation, and does not imply the order of transmission of the cell. The transmission of the cell is per the ATM standard.] Reflecting this byte-swap, the offset column gives the offset in MURAM from the ICP template base.

Table 35-9. ICP Cell Template [1]

Offset	Name	Width	Description
0x00	**ICP CELL HEADER**	Word	ICP cell header. Program to 0xD0000000.
0x04	ICP Cell Offset	Byte	Microcode-managed area. Microcode will program this field dynamically.
0x05	IMA Frame Sequence Number	Byte	Microcode-managed area. Microcode will program this field dynamically.
0x06	Cell ID and Link ID	Byte	Microcode-managed area. Microcode will program this field dynamically.
0x07	**OAM LABEL**	Byte	IMA Version value. 1 IMA Version 1.0 3 IMA Version 1.1

Table 35-9. ICP Cell Template (continued)[1]

Offset	Name	Width	Description
0x08	**GROUP STATUS AND CONTROL**	Byte	Bits 7-4: Group State 0000 = Start-up, 0001 = Start-up-Ack, 0010 = Config-Aborted - Unsupported M, 0011 = Config-Aborted - Incompatible Group Symmetry, 0100 = Config-Aborted - Unsupported IMA Version, 0101, 0110 = Reserved for other Config-Aborted reasons in a future version of the IMA specification, 0111 = Config-Aborted - Other reasons, 1000 = Insufficient-Links, 1001 = Blocked, 1010 = Operational, Others: Reserved for later use in a future version of the IMA specification. Bits 3-2: Group Symmetry Mode 00 = Symmetrical configuration and operation, 01 = Symmetrical configuration and asymmetrical operation (optional), 10 = Asymmetrical configuration and asymmetrical operation (optional), 11 = Reserved Bits 1-0: IMA Frame Length (00: M=32, 01: M=64, 10: M=128, 11: M=256)
0x09	**IMA ID**	Byte	Bits 7-0: IMA ID
0x0A	**STATUS AND CONTROL CHANGE INDICATION (SCCI)**	Byte	Software must increment this field in the new ICP template whenever a new ICP template is created.
0x0B	Link Stuff Indication	Byte	Microcode-managed area. Microcode will program this field dynamically.
0x0C	**RX TEST PATTERN**	Byte	Bits 7-0: Rx Test Pattern (value from 0 to 255)
0x0D	**TX TEST PATTERN**	Byte	Bits 7-0: Tx Test Pattern (value from 0 to 255)
0x0E	**TX TEST CONTROL**	Byte	Bits 7-6: Unused and set to 0 Bit 5: Test Link Command (0: inactive, I: active) Bits 4-0: Tx LID of test link (0 to 31)
0x0F	**TRANSMIT TIMING INFORMATION**	Byte	Bits 7-6: Unused and set to 0 Bit 5: Transmit Clock Mode: (0: ITC mode, 1: CTC mode) Bits 4-0: Tx LID of the timing reference (0 to 31)
0x10	**LINK 3 INFO**	Byte	Status and control of link with LID = 3
0x11	**LINK 2 INFO**	Byte	Status and control of link with LID = 2
0x12	**LINK 1 INFO**	Byte	Status and control of link with LID = 1
0x13	**LINK 0 INFO**	Byte	Status and control of link with LID = 0
0x14	**LINK 7 INFO**	Byte	Status and control of link with LID = 7
0x15	**LINK 6 INFO**	Byte	Status and control of link with LID = 6
0x16	**LINK 5 INFO**	Byte	Status and control of link with LID = 5
0x17	**LINK 4 INFO**	Byte	Status and control of link with LID = 4
0x18	**LINK 11 INFO**	Byte	Status and control of link with LID = 11
0x19	**LINK 10 INFO**	Byte	Status and control of link with LID = 10

Table 35-9. ICP Cell Template (continued)[1]

Offset	Name	Width	Description
0x1A	**LINK 9 INFO**	Byte	Status and control of link with LID = 9
0x1B	**LINK 8 INFO**	Byte	Status and control of link with LID = 8
0x1C	**LINK 15 INFO**	Byte	Status and control of link with LID = 15
0x1D	**LINK 14 INFO**	Byte	Status and control of link with LID = 14
0x1E	**LINK 13 INFO**	Byte	Status and control of link with LID = 13
0x1F	**LINK 12 INFO**	Byte	Status and control of link with LID = 12
0x20	**LINK 19 INFO**	Byte	Status and control of link with LID = 19
0x21	**LINK 18 INFO**	Byte	Status and control of link with LID = 18
0x22	**LINK 17 INFO**	Byte	Status and control of link with LID = 17
0x23	**LINK 16 INFO**	Byte	Status and control of link with LID = 16
0x24	**LINK 23 INFO**	Byte	Status and control of link with LID = 23
0x25	**LINK 22 INFO**	Byte	Status and control of link with LID = 22
0x26	**LINK 21 INFO**	Byte	Status and control of link with LID = 21
0x27	**LINK 20 INFO**	Byte	Status and control of link with LID = 20
0x28	**LINK 27 INFO**	Byte	Status and control of link with LID = 27
0x29	**LINK 26 INFO**	Byte	Status and control of link with LID = 26
0x2A	**LINK 25 INFO**	Byte	Status and control of link with LID = 25
0x2B	**LINK 24 INFO**	Byte	Status and control of link with LID = 24
0x2C	**LINK 31 INFO**	Byte	Program to 0x00.
0x2D	**LINK 30 INFO**	Byte	Status and control of link with LID = 30
0x2E	**LINK 29 INFO**	Byte	Status and control of link with LID = 29
0x2F	**LINK 28 INFO**	Byte	Status and control of link with LID = 28
0x30	CRC10	Hword	Microcode-managed area. Microcode will program this field dynamically.
0x32	**END-TO-END CHANNEL**	Byte	Program to any value desired for end-to-end implementation-dependent signaling, or program to 0x00 if unused.
0x33	**UNUSED**	Byte	Program to 0x6A.
0x34	**TAG**	Byte	Internally-used tag value indicating that this is an ICP cell. Program to 0x80.
0x35	**Reserved**	11 Bytes	Initialize to 0.

1 **Boldfaced** entries in the above table indicate parameters which must be initialized by the user. All other parameters are managed by the microcode and should be initialized to zero unless otherwise stated.

35.4.4.2 IMA Group Receive Table Entry

Table 35-10. IMA Group Receive Table Entry [1]

Offset	Name	Width	Description
0x00	**IGRCNTL**	Byte	IMA group receive control parameters.
0x01	IGRSTATE	Byte	IMA group receive state. Microcode-managed parameter. Initialize to zero at group startup.
0x02	**RIMAID**	Byte	Receive IMA ID. Program to the validated receive IMA ID value.
0x03	**IMAVER**	Byte	IMA version. Program to the validated IMA version value. 1 IMA Version 1.0 3 IMA Version 1.1
0x04	**RM**	Byte	Receive IMA frame size. Program to 31, 63, 127, or 255 for frame sizes of 32, 64, 128, or 256, respectively.
0x05	**RNUMLINKS**	Byte	Number of receive links in the IMA group that are in the active state (ILRCNTL[RXSC]=01.
0x06	DCB_RM	Byte	Current cell (x out of RM cells in a Frame) being processed by the reconstruction routine. Microcode-managed parameter. Initialize to zero at group startup.
0x07	RSCCI	Byte	Receive IMA SCCI field. Microcode-managed parameter. Holds the SCCI value of the last ICP cell received by this group.
0x08	**DCBLNK**	Hword	Pointer to link entry in group order table currently in use. Microcode-managed parameter. Initialize to value of RGRPORDER0 at group startup.
0x0A	DCBX	Hword	Delay-compensation buffer extraction pointer. Microcode-managed parameter. Initialize to zero at group startup.
0x0C	STALL_COUNT	Byte	Number of On-demand requests made for which there were no cells available in a link's DCB. Microcode managed parameter. Initialize to zero at group startup.
0x0D	REF_IFSN	Byte	Identifies the IFSN of the frame being processed by the "reconstruction" routine. Microcode managed parameter.
0x0E	GFP	Hword	Pointer to the start of the current frame being processed by the "reconstruction" routine. Microcode managed parameter. Initialize to zero at group startup.
0x10	**RGRPORDER0**	Hword	Offset of receive group order table 0 in MURAM.
0x12	**RGRPORDER1**	Hword	Offset of receive group order table 1 in MURAM.
0x14	**ALPHABETA**	Byte	Alpha and beta parameters for IMA frame synchronization mechanism (IFSM). Bits 0-3: Alpha. Allowable range is 1-2; typical value is 2. Bits 4-7: Beta. Allowable range is 1-5; typical value is 2.
0x15	**GAMMA**	Byte	Gamma parameter for IMA frame synchronization mechanism (IFSM). Allowable range is 1-5; typical value is 1.
0x16–0x17	—	Hword	Reserved. Must be initialized to zero at group startup.
0x18	**REF_LINK**	Word	Bit array identifying which of the links (such as, PHY) in this group are enabled. Bit 0 corresponds to PHY 0, bit 30 corresponds to PHY 30. Software must set the corresponding bits to 1 so that the delay compensation process for the link(s) is started.

Table 35-10. IMA Group Receive Table Entry (continued)[1]

Offset	Name	Width	Description
0x1C	LINK_ICP	Word	Bit array identifying which of the links in this group has received an ICP cell. A “1” in the corresponding link’s bit position indicates that an ICP cell has been received. Microcode-managed parameter. Initialize to zero at group startup.
0x20	LINK_DCB	Word	Bit array identifying which of the links in this group has started storing cells to its corresponding DCB. A “1” in the corresponding link’s bit position indicates that cells have been stored in the DCB. Microcode-managed parameter. Initialize to zero at group startup.
0x24	LINK_LD	Word	Bit array identifying which of the “added” links in this group has a Longer propagation Delay (LD) than any of the existing links. A “1” in the corresponding link’s bit position indicates that it has a longer propagation delay. Microcode-managed parameter. Initialize to zero at group startup.
0x28	—	Byte	Reserved. Must be initialized to zero at group startup.
0x29	**RVPHYNUM**	Byte	Receive Virtual PHY number. Maps this IMA receive group to a virtual PHY number for the purpose of address mapping of received cells. Note that this parameter must be unique; it must not conflict with the PHY number of any non-IMA PHYs or with the RVPHYNUM of any other IMA group. If there are any PHY numbers which will never be used in a system (such as the actual PHY with the address does not exist), then RVPHYNUM should be selected from one of these PHYs. If no such PHY numbers are available (such as the multi-PHY interface consists of the full 31 PHYs), then select RVPHYNUM from one of the PHY numbers of the PHYs within this IMA group.
0x2A	**STALL_THR**	Byte	Stall threshold. Used to detect stalled links when performing round-robin cell extraction from the delay compensation buffers (Dcbz). This is the number of cells which may be received without advancing the cell extraction pointer. The value is application-dependent and must be tuned by the user to the “expected worst case.” Its value depends on the depth of queues and FIFOs in the complete transmit/receive path, and the ‘burstiness’ of the behavior of the FIFOs. Assuming very bursty FIFOs, it is approximately: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO) where: a) RNUMLINKS is the number of links in the receive group order structure (regardless of link status). b) RX_FIFO is the depth of the receive FIFOs of the TC layer. c) 3 is the rounded value of the allowed transmit skew between links of a group (2.5, per the IMA standard). An optimal value for STALL_THR will be great enough to produce no link stall events in normal operation, but low enough to detect a failed link as quickly as possible.
0x2B	**IRGFS**	Byte	IMA Receive Group Frame Size Bits 0-5: Reserved Bits 6-7 - GSC_M: Value of received ICP cell Group Status Contl field (Bits 1:0) which determine the IMA frame size. This field must be programmed before links in the group are assigned
0x2C	—	Word	Reserved. Must be initialized to zero at group startups.

Table 35-10. IMA Group Receive Table Entry (continued)[1]

Offset	Name	Width	Description
0x30	LINK_DCBO	Word	Link DCB overflow interrupt indication. Bit array identifying which links have issued a link DCB overflow (DCBO) interrupt. This parameter ensures that only one DCBO interrupt is generated per event. Microcode managed parameter. Initialize to zero at group startup.
0x34–0x3F	—	3 Words	Reserved. Must be initialized to zero at group startups.

1 **Boldfaced** entries indicate parameters that must be initialized by the user. All other parameters are managed by the microcode and should be initialized to zero unless otherwise stated.

35.4.4.2.1 IMA Group Receive Control (IGRCNTL)

The fields of the IGRCNTL register are shown in Figure 35-18.

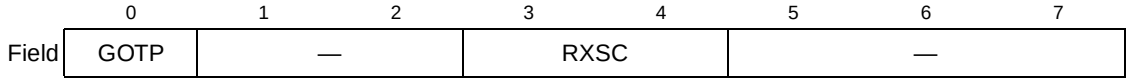


Figure 35-18. IMA Group Receive Control (IGRCNTL)

Table 35-11 describes the IGRCNTL bit fields.

Table 35-11. IGRCNTL Field Descriptions

Bits	Name	Description
0	GOTP	Group order table pointer. Defines which group order table pointer (RGRPORDER0 or RGRPORDER1) will be used for the cell extraction round-robin. Initialize to zero at group startup, such as, when programming RGRPORDER0 table, which must be used for the GDS process.
1–2	—	Reserved, initialize to zero.
3–4	RXSC	Receive status/control. Sets the receive mode of the IMA group. 00 Filler mode. The IMA group processes only ICP cells. Data cells are replaced with filler cells. 01 Active mode. The IMA group is capable of receiving data cells. 1X Reserved. Defaults to Filler Mode.
5–7	—	Reserved, initialize to zero.

35.4.4.2.2 IMA Group Receive State (IGRSTATE)

The fields of the IGRSTATE register are shown in Figure 35-19.

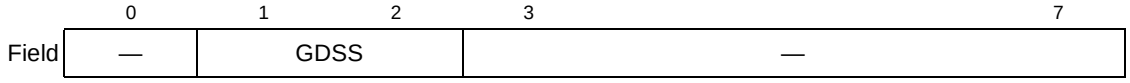


Figure 35-19. IMA Group Receive State (IGRSTATE)

Table 35-12 describes the IGRSTATE bit fields.

Table 35-12. IGRSTATE Field Descriptions

Bits	Name	Description
0	—	Reserved, initialize to zero.
1–2	GDSS	Group delay synchronization state. Initialize to zero at group startup. Must be changed by software to 01 after sufficient links have achieved frame synchronization. Subsequently managed by microcode. 00 Group delay synchronization process inhibited. 01 Group delay synchronization process enabled. 10 Group delay synchronization process in progress. 11 Group delay synchronized. Refer to Section 35.5.3.10, “Receive Event Response Procedures.”
3–7	—	Reserved, initialize to zero.

35.4.4.2.3 IMA Receive Group Frame Size

The fields of the IRGFS register are shown in Figure 35-20.

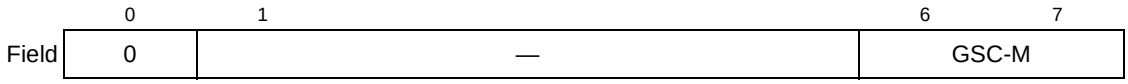


Figure 35-20. IMA Receive Group Frame Size (IGRSTATE)

Table 35-13 describes the IRGFS bit fields.

Table 35-13. IRGFS Field Descriptions

Bits	Name	Description
0	0	Reserved, initialize to zero.
1–5	—	Reserved, initialize to zero.
6–7	GSC-M	IMA Receive Group Frame Size Bits 0-5: Reserved Bits 6-7 - GSC_M: Value of received ICP cell Group Status Contl field (Bits 1:0) which determine the IMA frame size. This field must be programmed before links in the group are assigned

35.4.4.2.4 Receive Group Order Tables

The receive group order tables define the order of the links in the round-robin extraction of cells to the links of the IMA group. The table consists of an array of bytes ordered from first link to last link, with each byte providing the PHY address of its associated link. The end of the table is indicated by an entry programmed to 0x1F.

Two group order tables are used in order to allow for on-the-fly changes (such as to add or remove links), while making certain that the changes only occur at the end of a round-robin cycle. IGRCNTL[GOTP] indicates which group order table pointer (and therefore, group table) is currently in use. Changes should be made to the group order table not in use, and then IGRCNTL[GOTP] should be toggled. When the current round-robin cell extraction process completes, the next process will use the new table.

This table alone defines the order of cell extraction from the delay compensation buffers. It is the responsibility of software to read the LIDs of the links in the group from the received ICP cells during group startup, and to program the group order table such that its order corresponds to the order of increasing LIDs within the group.

The format of a receive group order table entry is shown in Figure 35-21.

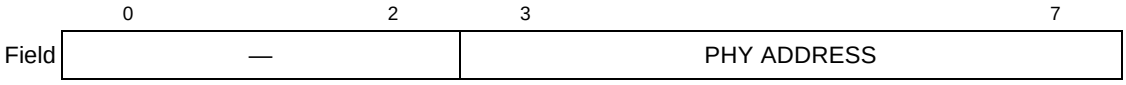


Figure 35-21. Receive Group Order Table Entry

Table 35-14 describes the format of a receive group order table entry.

Table 35-14. Receive Group Order Table Entry Field Descriptions

Bits	Name	Description
0–2	—	Reserved, initialize to zero.
3–7	PHY ADDRESS	PHY address (up to 31 PHYs[0–30]) of the link transmitting in this position in the round-robin. A value of 0x1F in this field indicates the end of the group order table. The value programmed must be the compressed value after Utopia Address Compression see Section 35.4.3.2, “Utopia PHY Address Compression.”

35.4.5 IMA Link Tables

The IMA link tables consist of multiple IMA group structures indexed by the PHY address of their corresponding PHYs. The transmit and receive parameters are located in separate tables. The IMA group transmit and receive table entries are each 32 bytes long. To conserve memory space consumed by these tables, it is best to group the addresses of the PHYs that are assigned as IMA. For example, if out of eight total links only four are IMA, then optimally these would be links 0–3.

35.4.5.1 IMA Link Transmit Table Entry

Table 35-15. IMA Link Transmit Table Entry [1]

Offset	Name	Width	Description
0x00	**ILTCNTL**	Byte	IMA link transmit control parameters.
0x01	ILTSTATE	Byte	IMA link transmit state. Microcode-managed parameter. Initialize to zero at link startup.
0x02	**LICPOS**	Byte	Link ICP offset. Determines the position of the ICP cell within the IMA frame. Program in the range 0 to M-1. See the IMA specification for recommended methods for programming this field.
0x03	**ILID**	Byte	IMA link ID. Formatted per the Cell ID and Link ID field of an ICP cell. Bit 0: 1 (indicating ICP cell) Bits 1-2: Not Used, set to zero Bits 3-7: Program to software-assigned LID. [Note: This value is only used by microcode to format ICP cells for this link, it is not used to determine round-robin transmission order. That function is performed by the group order table.]

Table 35-15. IMA Link Transmit Table Entry (continued)[1]

Offset	Name	Width	Description
0x04	ITSEC	Word	IMA transmit stuff event counter. While ILTCNTL[SES]=0, increments each time a stuff event is performed on this link. Initialize to zero at link startup.
0x08	**ITQSP**	Hword	IMA link transmit queue start pointer. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Refer to Section 35.4.6.1, “Transmit Queues.”
0x0A	**ITQEP**	Hword	IMA link transmit queue end pointer. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Points to the last cell in the transmit queue. Program this parameter to ITQSP+TQ_SIZE-4. Refer to Section 35.4.6.1, “Transmit Queues.”
0x0C	**ITQFP**	Hword	IMA link transmit queue fill pointer. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Initialize this to the value of ITQSP. Refer to Section 35.4.6.1, “Transmit Queues.”
0x0E	**ITQXP**	Hword	IMA link transmit queue extract pointer. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Initialize this to the value of ITQSP. Refer to Section 35.4.6.1, “Transmit Queues.”
0x10	**ITINTMSK**	Byte	IMA transmit interrupt mask. Has the same format as the upper byte of the IMA interrupt queue entry. Setting a bit enables the associated interrupt; clearing a bit masks it. For group-related events, only the mask register for the TRL is referenced.
0x11	ITINTSTAT	Byte	IMA transmit interrupt status. Indicates the status of transmit interrupt events.
0x12	LSHC	Byte	Stuff holdoff counter. Maintains the minimum five-frame spacing between stuff events for non-TRL links. Must be initialized to zero. Set to 4 by the microcode after a stuff event, and decrements after each ICP cell is sent (saturating at zero). Signaling of ‘imminent stuff’ (and subsequent stuff events) will not occur while this counter is non-zero.

1 **Boldfaced** entries indicate parameters that must be initialized by the user. All other parameters are managed by the microcode and should be initialized to zero unless otherwise stated.

35.4.5.1.1 IMA Link Transmit Control (ILTCNTL)

The fields of the ILTCNTL register are shown in Figure 35-22.

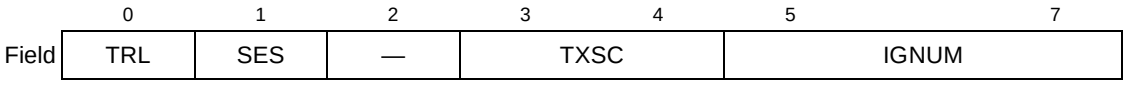


Figure 35-22. IMA Link Transmit Control (ILTCNTL)

Table 35-16 describes the ILTCNTL bit fields.

Table 35-16. ILTCNTL Field Descriptions

Bits	Name	Description
0	TRL	Defines this link as the timing reference link (TRL) of the group. 0 This link is not the TRL. 1 This link is the TRL. Note: One and only one link of the group must be programmed as the TRL. If zero or more than one links are defined as TRL, erratic operation will occur.
1	SES	Severely errorred seconds. Set by software when a severely errorred second indication is detected. When set, causes the transmit stuff event counter to stop counting.
2	—	Reserved, initialize to zero.
3–4	TXSC	Transmit status/control. Sets the transmit mode of the IMA link. 00 Filler mode. The IMA link transmits only ICP and filler cells, no data cells. 01 Active mode. The IMA link is capable of sending data cells. 1X Reserved.
5–7	IGNUM	Determines to which IMA group this link belongs. Contains the number of the link's associated IMA Group Table entry. Note: This value need not be the same as the group's IMA ID; the IMA ID is programmed separately in the group's ICP cell template.

35.4.5.1.2 IMA Link Transmit State (ILTSTATE)

The fields of the ILTSTATE register are shown in Figure 35-23

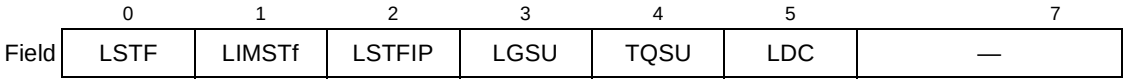


Figure 35-23. IMA Link Transmit State (ILTSTATE)

Table 35-17 describes the ILTSTATE bit fields.

Table 35-17. ILTSTATE Field Descriptions

Bits	Name	Description
0	LSTF	Link stuff flag. Microcode-managed parameter. Indicates that the next ICP cell on this link will be part of a stuff event. Initialize to zero at link startup.
1	LIMSTF	Link imminent stuff flag. Microcode-managed parameter. Indicates that an upcoming stuff event will be signaled in the next ICP of this link (via LSI=001). Initialize to zero at link startup.
2	LSTFIP	Link stuff-in-progress flag. Microcode-managed parameter. Indicates that the next cell on this link will be the second cell of a stuff event. Initialize to zero at link startup.
3	LGSU	Link/group startup flag. Microcode-managed parameter. Will be set by the microcode after the link's transmit queue has reached target average depth of 3 cells. While this bit is cleared, the link will transmit only filler cells. Initialize to zero at link startup.
4	TQSU	Transmit queue startup flag. Microcode-managed parameter. Will be set by the microcode after the first cell has been sent to the transmit queue. Initialize to zero at link startup.

Table 35-17. ILTSTATE Field Descriptions (continued)

Bits	Name	Description
5	LDC	Link Due Cell - Link is due cell from round robin distribution. Microcode managed parameter initialize to 0. Used for optional TRL Service Latency enhancement only.
6–7	—	Reserved, initialize to zero.

35.4.5.1.3 IMA Transmit Interrupt Status (ITINTSTAT)

The fields of the ITINTSTAT register are shown in Figure 35-24.

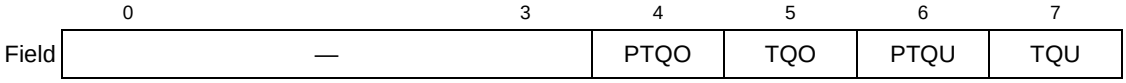


Figure 35-24. IMA Transmit Interrupt Status (ITINTSTAT)

Table 35-18 describes the ITINTSTAT bit fields.

Table 35-18. ITINTSTAT Field Descriptions

Bits	Name	Description
0–3	—	Reserved, initialize to zero.
4	PTQO	Persistent transmit queue overflow. Set when a transmit queue overflow occurs for two cells in a row. Further transmit queue overflow events are masked when this bit is set, in order to avoid a ‘flood’ of interrupts. This bit can be used to distinguish a temporary overflow condition (which could be caused by a rate differential between the TRL and non-TRL link which was out of compensatable range), or a persistent overflow condition (which could be caused by a more permanent condition, such as a failure of this link’s PHY device). Initialize to zero at link startup.
5	TQO	Transmit queue overflow. Set when a transmit queue overflow occurs; cleared when a cell is successfully transmitted. As this bit is set or cleared on a cell-by-cell basis, it may no longer be set when software reads it if the overflow condition was only temporary. PTQO should be used instead to distinguish between persistent or temporary underrun conditions.
6	PTQU	Persistent transmit queue underrun. Set when a transmit queue underrun occurs for two cells in a row. Further transmit queue underrun events are masked when this bit is set, in order to avoid a ‘flood’ of interrupts. This bit can be used to distinguish a temporary underrun condition (which could be caused by a rate differential between the TRL and non-TRL link which was out of compensatable range), or a persistent underrun condition (which could be caused by a more permanent condition, such as a TRL failure). Initialize to zero at link startup.
7	TQU	Transmit queue underrun. Set when a transmit queue underrun occurs; cleared when a cell is successfully transmitted. As this bit is set or cleared on a cell-by-cell basis, it may no longer be set when software reads it if the underrun condition was only temporary. PTQU should be used instead to distinguish between persistent or temporary underrun conditions. Initialize to zero at link startup.

35.4.5.2 IMA Link Receive Table Entry

Table 35-19. IMA Link Receive Table Entry[1]

Offset	Name	Width	Description
0x00	**ILRCNTL**	Hword	IMA link receive control parameters.
0x02	ILRSTATE	Hword	IMA link receive state. Microcode-managed parameter. Initialize to 0x0040 at link startup.
0x04	**ILID**	Byte	IMA link ID. Formatted per the Cell ID and Link ID field of an ICP cell. Bit 0: 1 (indicating ICP cell) Bits 1-2: Program to zero Bits 3-7: Program to validated LID for this link [Note: This value is only used by microcode to validate incoming ICP cells for this link, it is not used to determine round-robin transmission order. That function is performed by the group order table.]
0x05	RMCTR	Byte	Receive M counter. Microcode-managed parameter.
0x06	LRIFSN	Byte	Receive IFSN counter. Microcode-managed parameter.
0x07	DFC	Byte	Number of frames to discard on a this link until it is caught up with the other links in this group (long propagation delay). Microcode-managed parameter.
0x08	**DCBSP**	Hword	IMA link delay compensation buffer start pointer. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Refer to Section 35.4.6.2, “Delay Compensation Buffers (DCB)” for more details.
0x0A	**DCBEP**	Hword	IMA link delay compensation buffer end pointer. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Refer to Section 35.4.6.2, “Delay Compensation Buffers (DCB)” for more details.
0x0C	**DCBFP**	Hword	IMA link delay compensation buffer fill pointer. Microcode-managed parameter. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero. Initialize to DCBSP at link startup.
0x0E	**DCBRP**	Hword	IMA link delay compensation buffer read pointer. Only used during group delay synchronization and link addition. Microcode-managed parameter. This parameter forms bits 12-27 of the pointer; bits 0-11 are from IMAEXTBASE, and bits 28-31 are zero.
0x10	**RICPCH**	Hword	Receive ICP channel number. ATM receive channel number to which received ICP cells are sent.
0x12	**IRINTMSK**	Byte	IMA receive interrupt mask. Has the same format as the IMA interrupt queue entry.; however, only receive-related bits are relevant. Setting a bit enables the associated interrupt; clearing a bit masks it. For group-related events, only the mask register for the TRL is referenced.
0x13	**LICPOS**	Byte	Link ICP offset. Program to the ICP offset validated for this link.
0x14	IRSEC	Word	IMA receive stuff event counter. Increments each time a stuff event is received on this ink. Initialize to zero at link startup.
0x18	ANOMALY_CTR	Byte	Anomaly counter. Microcode-managed parameter. Initialize to zero at link startup.

Table 35-19. IMA Link Receive Table Entry[1] (continued)

Offset	Name	Width	Description
0x19	ALPHABETA_CTR	Byte	Alpha/beta counter. Microcode-managed parameter. Initialize to zero at link startup.
0x1A	GAMMA_CTR	Byte	Gamma counter. Microcode-managed parameter. Initialize to zero at link startup.
0x1B	—	Byte	Reserved. Must be initialized to zero.
0x1C	DEFECT_CTR	Hword	Defect counter. This counter is active while the link is in the Loss-of-IMA-Frame (LIF) state and is used to ensure IFSD interrupts are generated for every GAMMA+2 frames. Software can use the period interrupt issued by this counter in order to determine if the link is taking too long to synchronize. The DEFECT_CTR is active before IFSM reaches SYNC. It starts counting from the first cell received and will count from 0 to (GAMMA+2) x M. When it reaches (GAMMA+2) x M an IFSD interrupt is generated and the counter is reset. Upon reception of the next cell it starts to count again and subsequent interrupts are generated. Microcode managed parameter. Initialize to zero at link startup.
0x1E	—	Hword	Reserved. Must be initialized to zero.

1 **Boldfaced** entries indicate parameters that must be initialized by the user. All other parameters are managed by the microcode and should be initialized to zero unless otherwise stated.

35.4.5.2.1 IMA Link Receive Control (ILRCNTL)

The fields of the ILRCNTL register are shown in Figure 35-25

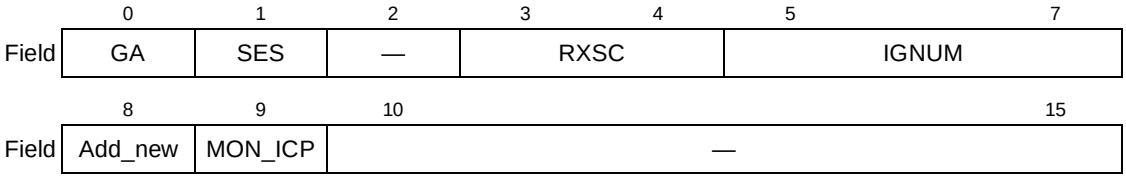


Figure 35-25. IMA Link Receive Control (ILRCNTL)

Table 35-20 describes the ILRCNTL bit fields.

Table 35-20. ILRCNTL Field Descriptions

Bits	Name	Description
0	GA	Group assigned flag. Set by software after group parameters have been validated. 0 Group unassigned. 1 Group assigned.
1	SES	Severely errorred seconds. Set by software when a severely errorred second indication is detected. When set, causes the receive stuff event counter and ICP violation counter to stop counting.
2	—	Reserved, initialize to zero.

Table 35-20. ILRCNTL Field Descriptions (continued)

Bits	Name	Description
3–4	RXSC	Receive status/control. Sets the receive mode of the IMA link. 00 Filler mode. The IMA link processes only ICP cells. Data cells are replaced with filler cells. 01 Active mode. The IMA link is capable of receiving data cells. 10 Dropped. Must be set by the user during the process of dropping the link. Dropped links are treated filler mode until they are switched out of the receive round-robin (such as data cells are replaced with filler cells). 11 Reserved.
5–7	IGNUM	Determines to which IMA group this link belongs. Contains the number of the link's associated IMA Group Table entry. Note: This value need not be the same as the group's IMA ID; the IMA ID is programmed separately in the receive group's RIMAID field.
8	ADD_NEW	Identifies this link as a new/added link to a group. Software must flip (0 to 1 or 1 to 0) this bit ONLY when adding a link to an existing group that is operational. If ADD_NEW = ILRSTATE[ADD_NEW_M] = 0, set ADD_NEW to 1 to indicate this is an added link. If ADD_NEW = ILRSTATE[ADD_NEW_M] = 1, set ADD_NEW to 0 to indicate this is an added link. Initialize to zero.
9	MON_ICP	Setting this bit will allow changed ICP cells received on this link to be passed on to the ATM layer (defined channel). The user must monitor at least one link in order for ICP cells to be passed on to the ATM layer.
10–15	—	Reserved, initialize to zero.

35.4.5.2.2 IMA Link Receive State (ILRSTATE)

The fields of the ILRSTATE register are shown in Figure 35-26

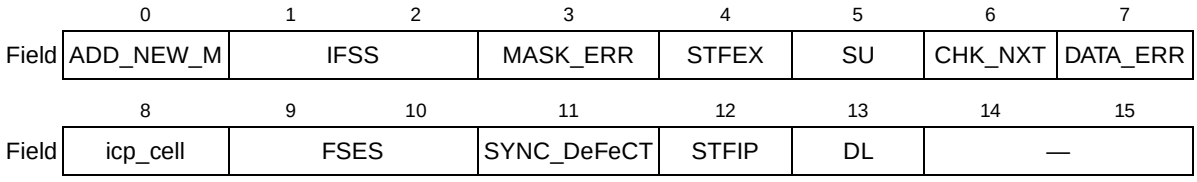


Figure 35-26. IMA Link Receive State (ILRSTATE)

Table 35-21 describes the ILRSTATE bit fields.

Table 35-21. ILRSTATE Field Descriptions

Bits	Name	Description
0	ADD_NEW_M	'New Link' shadow bit. Microcode managed parameter. Initialize to zero at link startup.
1–2	IFSS[1]	IMA frame synchronization state. Microcode-managed parameter. Initialize to zero at link startup. 00 IMA hunt. 01 IMA presync. 1x IMA sync.
3	MASK_ERR	Mask Error. Microcode managed parameter. Initialize to zero at link startup.
4	STFEX	Stuff cell expected. Microcode-managed parameter. Initialize to zero at link startup.
5	SU	Start Up. Microcode-managed parameter. Initialize to zero at link startup.

Table 35-21. ILRSTATE Field Descriptions (continued)

Bits	Name	Description
6	CHK_NXT	Check Next. Microcode-managed parameter. Initialize to zero at link startup.
7	DATA_ERR	Data Error - Parity/CRC. Microcode-managed parameter. Initialize to zero at link startup.
8	ICP_CELL	Current Cell is an ICP Cell. Microcode-managed parameter. Initialize to zero at link startup.
9–10	FSES[1]	Frame Synchronization Error State. Microcode-managed parameter. Initialize to 10 at link startup. 00 IMA working. 01 Out-of-IMA frame anomaly. 1x Loss-of-IMA frame defect.
11	SYNC_DEFECT	Synchronization Defect. Microcode-managed parameter. Initialize to zero at link startup.
12	STFIP	Stuff In-Progress flag. Microcode-managed parameter. Initialize to zero at link startup.
13	DL	Dropped link. Microcode-managed parameter. Initialize to zero at link startup.
14–15	—	Reserved, initialize to zero.

[1]Enable these parameters to their default values during IMA initialization to avoid spurious IMA events in the IMA interrupt queues.

35.4.5.3 IMA Link Receive Statistics Table

The IMA link receive statistics table is optional. It is enabled globally for this UCC via IMACNTL[IRSE]. The base of this table is determined by IRLINKSTAT. Entries in the table are indexed by the link number of the associated link. The format for the IMA link receive statistic table entries is shown in Table 35-22

Table 35-22. IMA Link Receive Statistics Table Entry

Offset	Name	Width	Description
0x00	ICPVIOL	Word	IMA receive ICP violation event counter. While ILRCNTL[SES]=0, increments each time an errorred, invalid, or missing ICP cell is received on this link. Initialize to zero at link startup.
0x04	OIF	Word	Out-of-IMA frame counter. While ILRCNTL[SES]=0, increments each time an Out-of-IMA Frame anomaly occurs on this link. Initialize to zero at link startup.

35.4.6 Structures in External Memory

The IMA microcode requires supporting queue structures in external memory. These are used for the jitter buffers (on transmission) and the delay compensation buffers (on reception). These structures reside in a 1 megabyte memory region defined by IMAEXTBASE, and may reside on either the CSB bus or the secondary bus, as defined by IMACNTL[DSB].

35.4.6.1 Transmit Queues

Transmit queues are allocated on a per-link basis. They are circular queues of 64-byte buffers, defined by their start and end pointers. For the transmit queue of the timing reference link (TRL), the queue must consist of a minimum of four buffers (although it may consist of five buffers, if consistency of data structures is desired). For the transmit queues of non-TRL links, the queues must consist of five buffers.

Queue fill and extract pointers must be initialized by software to the start of the queue; thereafter, these pointers are managed by the microcode. The queue pointers must be on a 64-byte aligned boundary.

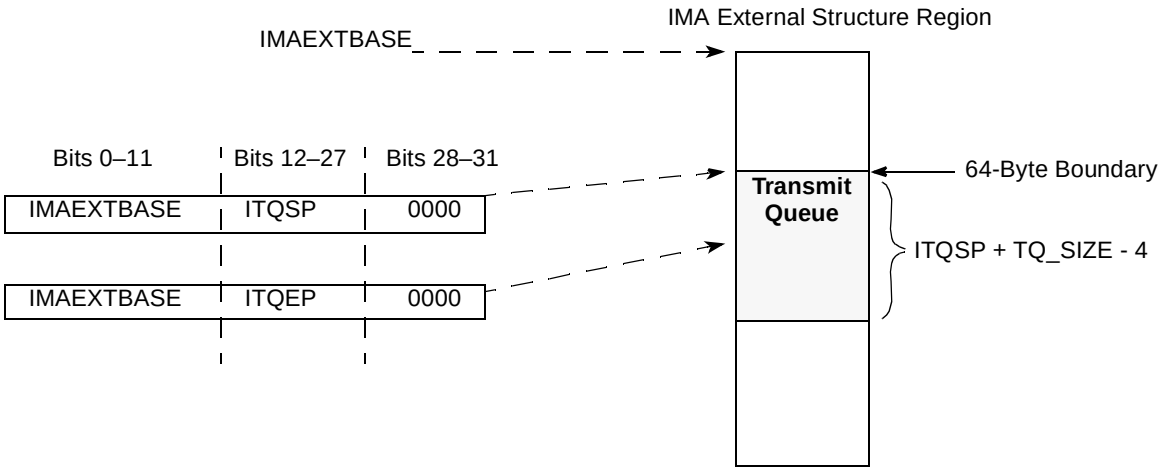


Figure 35-27. IMA Transmit Queue

35.4.6.2 Delay Compensation Buffers (DCB)

Cells received on a link are initially stored in a delay compensation buffer (DCB). DCBs are allocated on a per-link basis. They are of user-definable length and provide a programmable maximum synchronizable delay. DCBs of links within a group must all have the same size. DCBs consist of a circular queue of 64-byte cell buffers that contain the received cell followed by 12 bytes of header/status information.

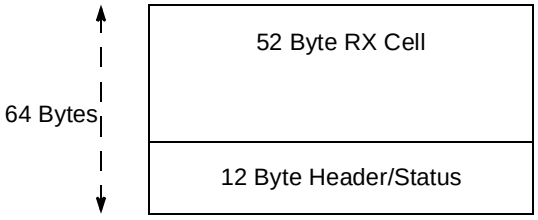


Figure 35-28. Cell Buffer in Delay Compensation Buffer

The length of a DCB is defined by the link DCBSP and DCBEP parameters. The length of the DCB (defined by (DCBEP–DCBSP) x 16) must be an integer multiple of the IMA frame length in bytes, M x 64. Furthermore, the DCBSP must be aligned on a M x 64-byte boundary. For example, if M = 64, DCBSP must be on a 4 Kbyte boundary. To ensure group delay synchronization, the minimum length of the DCB should be $2(M \times 64)$. The DCB memory area must be initialized to zero at link startup.

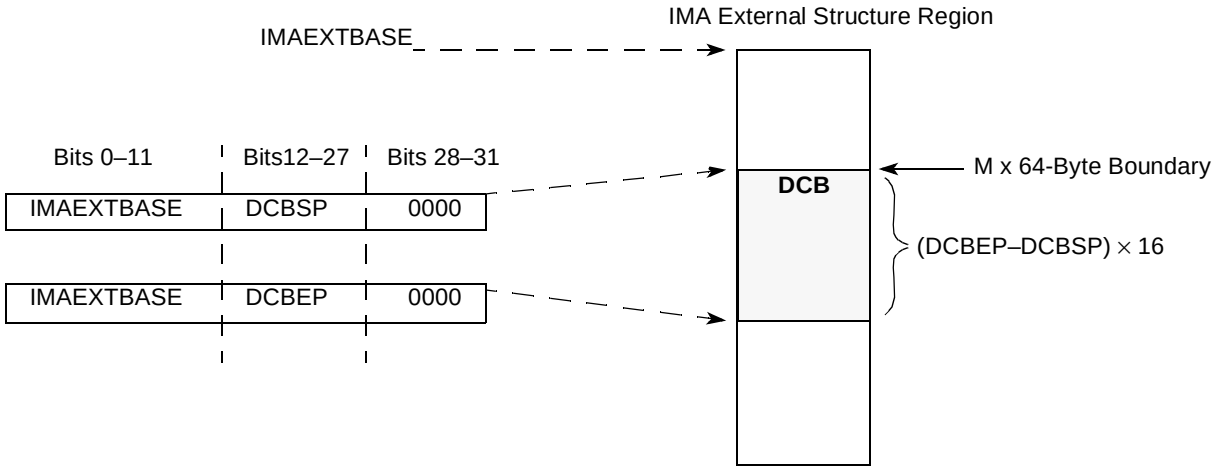


Figure 35-29. IMA Delay Compensation Buffer

35.4.7 IMA Events

One of the four ATM interrupt queues must be dedicated to IMA events. This requirement enables minimum latency in dealing with the IMA state machines and ensures that IMA events are not confused with other events. The IMA interrupt queue is defined in the IMACNTL[INTQ] parameter. IMA events sent to this queue include only those described in this section. ICP receive events are treated as normal receive events, and should therefore go to the interrupt queue allocated for receive events.

35.4.7.1 IMA Interrupt Queue Entry

The format for the IMA interrupt queue entries is shown in Figure 35-30

	0	1	2	3 5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	V	—	W	—	TQU	TQO	—	DSL	LS	DCBO	LDS	GDS	IFSD	IFSW
OFFSET + 2	L/G	NUM												

Figure 35-30. IMA Interrupt Queue Entry

Table 35-23 describes the IMA interrupt queue entry bit fields.

Table 35-23. IMA Interrupt Queue Entry Field Descriptions

Offset	Bits	Name	Description
Offset + 0	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	—	Reserved.
	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.
	3–4	—	Reserved
	5	—	Reserved
	6	TQU	Transmit queue underrun. Indicates that the corresponding PHY for this link requested a cell for transmission, but the transmit queue was empty.
	7	TQO	Transmit queue overflow. Indicates that the TRL attempted to send a cell to this link's transmit queue, but no space was available.
	8	—	Reserved
	9	DSL	DCB synchronization lost. This interrupt is issued when a link in a group with IGRSTATE[GDSS] = 11 loses synchronization and the link enters HUNT state at the IFSM.
	10	LS	Link stalled. A link in the round-robin cell extraction process has excessively stalled and been deactivated (such as switched to filler mode by the microcode).
	11	DCBO	Link out of delay synchronization. Set when the link's DCB overflows, indicating that delay synchronization for this link is not possible.
	12	LDS	Link delay synchronized. Set when delay synchronization is achieved for a link that has been added to an existing group.
	13	GDS	Group delay synchronized. Set when a group achieves delay synchronization as part of the group startup procedure.
	14	IFSD	IMA frame synchronization status = defect. Set when the associated link goes into the Loss of IMA Frame Defect state of the Error/Maintenance State Machine.
	15	IFSW	IMA frame synchronization status = working. Set when the associated link goes into the IMA Working state of the Error/Maintenance State Machine.
Offset + 2	0	L/G	Link/group indicator. Indicates whether this interrupt is associated with a link or a group, and thus if the NUM field is a link number or group number. 0 This interrupt is associated with a link. 1 This interrupt is associated with a group.
	1–15	NUM	Link or group number associated with this interrupt.

35.4.7.2 ICP Cell Reception Events

ICP cells are received as AAL0 in the channel defined in RICPCH. Receive interrupts are provided for this channel if enabled in its associated RCT.

35.4.8 APC Programming for IMA

Dynamically adding and dropping links from a group changes the overall bandwidth of the group. The bandwidth of a particular ATM channel is programmed as a percentage of the overall bandwidth, up to 100 percent for an APC pace of 1. For IMA, it is desirable to allow for the dynamic addition and deletion of links without changing the bandwidth of individual ATM channels so that ATM traffic contracts are not violated. To do this without requiring updates of the APC parameters of all of the ATM channels, the APC algorithm must be modified to consider the number of links in the group.

To accomplish this for channels to be used with IMA groups, the APC parameters should be programmed to define the bandwidth of a channel as a percentage of one link of the IMA group. The APC pace can be greater than 100 percent if a channel uses more bandwidth than a single link can provide. The APC pace is scaled automatically by the IMA group transmit parameter TNUMLINKS. After scaling, if the pace required of the group is still greater than 100 percent of the group bandwidth, the channel is rescheduled at 100 percent of the group bandwidth. Refer to the examples in Table 35-24.

Table 35-24. Examples of APC Programming for IMA

Example	Description
1	The simplest case. For an IMA group consisting of one 2Mbps link, with one CBR channel consuming the full bandwidth of that link (2 Mbps), TNUMLINKS for the group should be programmed to 1 and the APC pace of the channel should be programmed to 1 (PCR=1, PCR_Fraction=0).
2	Another 2 Mbps link is added to the IMA group in Example 1. The overall bandwidth of the group is now 4 Mbps. However, TNUMLINKS is now 2, and therefore the programmed PCR will be scaled by 2. [Note that the scaling is done automatically internally; the PCR programmed into the channels transmit connection table entry is not modified.] A scaled PCR of 2 indicates that the channel uses 50% of the bandwidth of the group, or 2Mbps. Comparing to example 1 above, we see that the bandwidth of the channel has not changed, even though the bandwidth of the group has changed.
3	Consider an IMA group consisting of five 2 Mbps links over which a 6 Mbps CBR channel is to be sent. 6 Mbps is 300 percent of what a single 2 Mbps link can provide, so its pace should be programmed as 1/3 (PCR=0, PCR_Fraction=85). Scaling the pace by TNUMLINKS results in 5/3 (PCR=1, PCR_Fraction=169), which indicates that the channel uses 60 percent of the bandwidth of the group, or 6 Mbps.
4	Suppose one link is dropped from the IMA group in Example 4. The overall bandwidth of the group is now 8 Mbps, and TNUMLINKS becomes 4. Therefore, the pace for the 6 Mbps CBR channel scales to 4/3 (PCR=1, PCR_Fraction=84), indicating that the channel uses 75 percent of the bandwidth of the group, which is still 6 Mbps.
5	Suppose two more links are dropped from the IMA group in Example 4. The overall bandwidth of the group is now 4 Mbps, and TNUMLINKS becomes 2. Therefore, the pace for the 6 Mbps CBR channel scales to 2/3 (PCR=0, PCR_Fraction=170). This is less than 1, which is not possible to support, so it is rounded up to 1. The channel originally programmed for 6 Mbps now consumes 100 percent of the 4 Mbps IMA group.

Per the above explanation and examples, TNUMLINKS is the only parameter that software must modify when a link is added or dropped from an IMA group. All other APC parameters need not be modified. However, if links are dropped so that the total scheduled bandwidth of the ATM channels is greater than 100 percent of the IMA group bandwidth, the eventual result is APC overruns, which should therefore be corrected and/or avoided.

NOTE

Software should ensure that the length of the APC scheduling table is increased if links are added to the IMA group. When incrementing the number of links in an IMA group, the user might exceed the length of the APC scheduling table; if this happens, the ATM channel is mapped outside of the APC scheduling table and the channel stops.

35.4.8.1 Programming for CBR, UBR, VBR, and UBR+

All APC parameters for CBR, UBR, VBR, and UBR+ channels to be transmitted over IMA groups should be scaled by the intended steady-state value of TNUMLINKS. Their values should be divided by TNUMLINKS because they will be scaled by TNUMLINKS when the pacing algorithms are performed. The parameters that must be scaled include: PCR, SCR, OOBR, BT, MCR, and MDA.

35.4.9 Changing IMA Version

A CECR command added to the IMA microcode changes the IMA version on-the-fly without software intervention. Use the following procedure:

1. Before issuing the command, initialize the COMM_INFO fields in the parameter RAM as shown in Figure 35-31.
2. To issue this UCC command, refer to Section 19.3.1.1, “QUICC Engine Commands.” Use opcode 001101(0x0D).

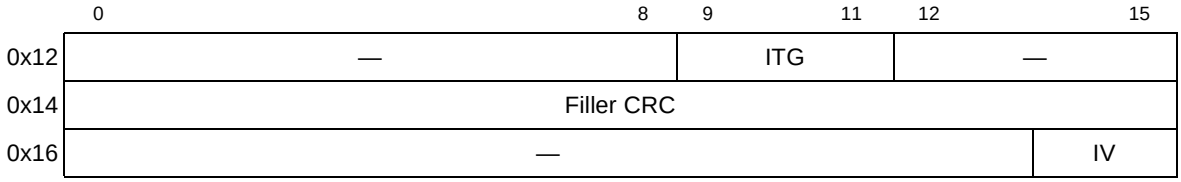


Figure 35-31. COMM_INFO Field

Table 35-25. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x12	0–8	—	Reserved, should be cleared.
	9–11	ITG	IMA transmit group. Program to the IMA group number [0–7] multiplied by 16 bytes. For example, if the IMA group number = 3, program ITG to 0x30.
	12–15	—	Reserved, should be cleared.
0x14	0–15	Filler CRC	Filler cell CRC. Set to 0xC602 (for IMA version 1.0) or 0xD902 (for IMA version 1.1)
0x16	0–13	—	Reserved, should be cleared.
	14–15	IV	IMA version 00 Reserved 01 IMA version 1.0 10 Reserved 11 IMA version 1.1

3. Wait for the QUICC Engine block to clear the CECR[FLG] before issuing a new CP command. Refer to Section 19.3.1.1, “QUICC Engine Commands.”

35.5 IMA Software Interface and Requirements

The IMA microcode facilitates the implementation of the lower levels of an IMA system. Host software, which from here on is code running on the G2 core, initializes the processor IMA data structures, establishes and tears down connections, handles alarms, keeps statistics, and controls protocol state machines. The IMA microcode interfaces to the software-implemented (layer management and plane management) functions by providing received ICP cells and by interrupts. The software-implemented functions control the microcode and the system through the IMA root, group, and link parameters and by providing an ICP cell template to the microcode for ICP cell transmission.

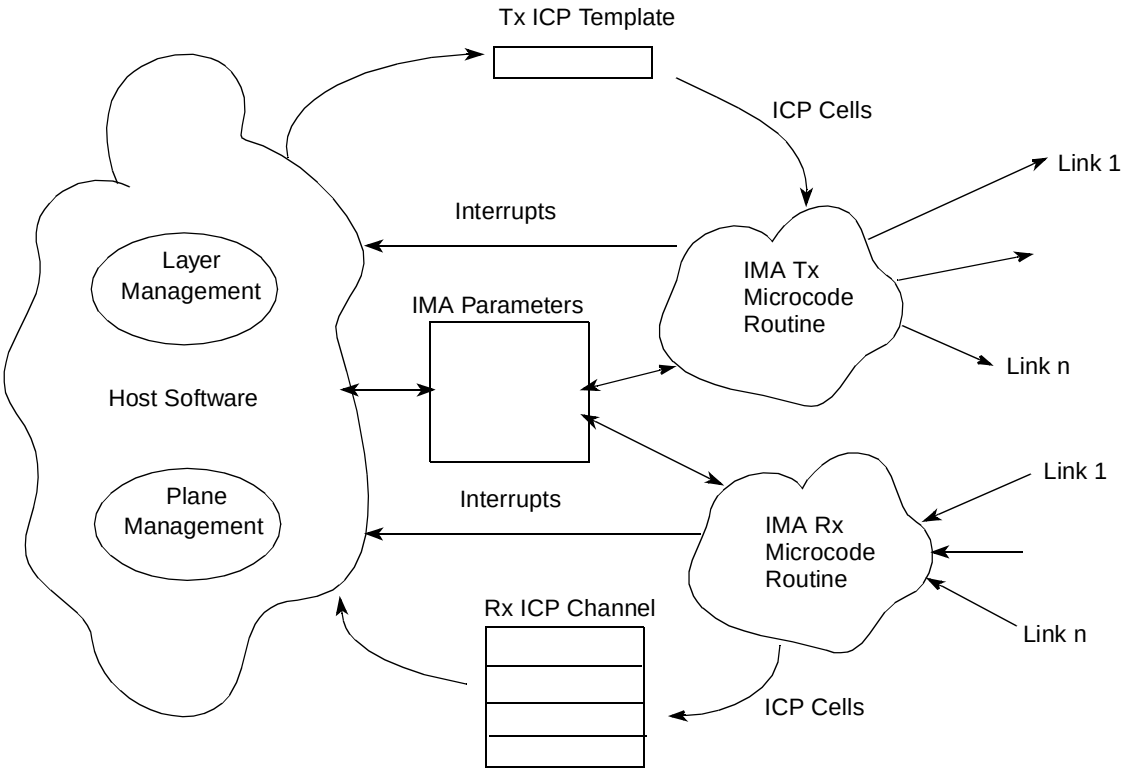


Figure 35-32. IMA Microcode/Software Interaction

35.5.1 Initialization Procedure

1. Program the UCC registers/parameters for ATM operation with UTOPIA multi-PHY (excluding APC parameters for IMA PHYs).
2. Program the IMA UCC and root parameters.
3. Enable the UCC through the GFMRx[ENR,ENT] bits.

Aside from IMA state machine control and IMA-specific error events, subsequent interaction with the ATM channels is the same as for non-IMA operation (for example, host commands, RCT/TCT parameters, buffer descriptors, interrupts).

35.5.2 Software Responsibilities

Table 35-26 lists functions that are the responsibility of the host software, which must complement the IMA microcode to provide a complete IMA solution.

Table 35-26. Host Software Functions

Function	Action
System definition	Define the P(Rx) and P(Tx) software variables to determine sufficient links.
General Operation	React to received ICP cells. ICP cells are received only when their SCCI field changes, except the first received ICP cell.
	Report any NE and FE timing mode mismatches (ITC versus CTC) to the unit management.
Receive link state machine control	Set up unassigned links, awaiting ICP cells: • Define either as standard multi-PHY or as unassigned IMA link. • Set up receive channel for ICP OAM cells for this link only.
	Create and initialize the delay compensation buffer.
	Respond to received ICP cells. Extract and validate link/group parameters: • IMA ID • Link ID • Link ICP offset • IMA frame size (M)
	After establishing the group, change the receive channel for ICP OAM cells so that all links in the group point to the same channel.
	Control link operation (through ILRCTNL[RXSC]) in coordination with link and group states.
Receive group state machine control	Coordinate receive links using received ICP cells to identify a proposed group.
	Establish and validate group parameters in conjunction with the received link states and parameters • IMA version • IMA ID • Group order table
	Enable delay synchronization for the group, and react to its completion.
	Control group operation (through IGRCNTL[RXSC]) in coordination with group state.
	Signal receive group state through ICP cells of corresponding transmit group.
Transmit link state machine control	Define the IMA link • Define link parameters (for example, IMA ID, Link ID, IMA frame size (M)) in coordination with IMA group definition • Assign ICP offset • Create and initialize transmit queue
	Control link operation (through ILTCNTL[TXSC]) in coordination with link and group states.

Table 35-26. Host Software Functions (continued)

Function	Action
Transmit group state machine control	Define the IMA group(s) • Assign IMA ID • Assign Link IDs and the transmit group order table • Assign TRL • Assign IMA frame size (M) • Signal group parameters via ICP cells • Define ATM pace controller (APC) parameters for this transmit group
	Signal transmit group state through ICP cells.
	Negotiate transmit group parameters with the far end.
	Check status of links in group to make ‘Sufficient Links’ determination.
	Control group operation (through IGTCNTL[TXSC]) in coordination with group state.
	Coordinate link state transitions with group state transitions during group startup and link addition.
Group symmetry control	Support for all types of group symmetry (operation and configuration) is facilitated by the ability to enable/disable links independently and to control link and group states through independent link and group transmit and receive parameters.
ICP end-to-end channel transmission	Can send information on end-to-end channel by updating field in Tx ICP.
	No Rx support is provided for end-to-end channel.
Link addition and slow recovery (LASR) procedure	Coordinate addition of links for both receive and transmit, including the following: • Establishing their parameters • Checking for defects • On-the-fly insertion into data structures
Failure alarms	React to errors signaled in received ICP cells.
	React to physical layer errors.
	Monitor persistence of errors to determine alarm condition.
	Report receive defects within 2M cells of entering defect state.
	Signal upper-layer software.
Test pattern control	Initiate transmit test patterns in the group’s transmit ICP cells (through TXTC and TXTP) and monitor the response in the receive ICP cells of the associated receive group.
	Respond to test patterns by: • Recognizing test pattern activity in the received ICP cells • Locating the Tx Test Pattern field in the ICP cell of the test link. (Because the SCCI field changes as part of the test pattern generation by the far end, the cell will necessarily be among the received ICP cells. Locate the latest ICP cell with a LID matching the Tx LID of the test link). • Signaling back via the transmit ICP cells of the associated transmit group, by modifying the transmit ICP cell template appropriately

Table 35-26. Host Software Functions (continued)

Function	Action
Performance parameter measurement and reporting	Establish timers to correlate errors with time intervals (for example, to determine severely errored seconds (SES) or unusable seconds (UUS))
	Maintain statistics.
	Enable/disable receive and transmit event counters according to severely errored seconds (SES) condition through ILRCNTL[SES] and ILTCNTL[SES].
SNMP MIBs	Control interface and statistics information should be provided per the SNMP MIB definition for IMA.

35.5.3 IMA Software Procedures

Procedures must be followed to assure the synchronization of changes between the link state machines and the group state machine. Issues to be considered are order of changes to the ICP cell and pointer, group order structure and pointer, IMA PHY assignment structure, and group/link Tx control fields.

35.5.3.1 Transmit ICP Cell Signaling

1. Copy the transmit ICP cell template currently in use (as indicated by TICPPTR) to the ICP cell template area not in use.
2. In the new template, change the ICP cell template fields as appropriate.
3. Verify that the IGTCNTL[ICPC] equals IGTSTATE[ICPCA]. If not, wait until it does.
4. Change TICPPTR to point to the “changed/altered” (see step 1) ICP cell template.
5. Toggle IGTCNTL[ICPC].

35.5.3.2 Transmit Group Initialization

1. Initialize all IMA parameters, this must include the Transmit Group Order Tables for the link
2. Enable the corresponding TC layer to request cells, basically stage 1 should be complete before the TC layer requests cells to be delivered. A request for cells is an assertion of TxClav by the TC layer when using an external TC layer on the UTOPIA bus. When using the SAM a request for cells would be setting MTC_MODE[TXEN]. The TRL link should be enabled last.

35.5.3.3 Receive Link Startup Procedure

Before links and group can be activated, ICP cells must be received and processed by the management software. Software must configure all IMA links to group unassigned mode. Reception of ICP cells requires that the corresponding PHYs (that is, links) be enabled and the corresponding connection table entry/entries initialized (see RXPHYEN_TPTR and RICPH). In group unassigned mode, the first received ICP cell is always reported to the corresponding channel’s buffer (RICPCH) and subsequently every time there is a change in the Status and Control Change Indication (SCCI) field of the ICP cell.

- Clear ILRCNTL[GA].

Clearing GA (group unassigned) allows only ICP cells to be processed; all other cells are dropped. The management software analyzes the ICP cells and programs the corresponding IMA link receive table parameters:

- IMA Link ID (ILID)
- Link ICP offset (LICPOS)
- Select link as the Timing Reference Link (only one link can be TRL). ILRCNTL[TRL] = 1.
- Assign the link to a group. ILRCNTL[IGNUM] = x.
- Verify that the size of frame (M) is the expected value.

The software must have built-in knowledge of the mapping between physical links and their corresponding channel number (for example, PHY 0 uses channel 2 (RICPH = 2)). After a group is established, the user can have all links in a group report changed ICP cells to a single channel either by changing RICPCH for all the links to be the same or by clearing the MON_ICP bit:

- ILRCNTL[MON_ICP] = 0. At least 1 link in the group must have this bit set.

35.5.3.4 Group Startup Procedure

The IMA frame synchronization mechanism (IFSM) is initiated when a link is switched to group assigned. Management software must already have programmed the group parameters based upon the received/negotiated values in ICP cells:

- IMA ID (RIMAID)
- IMA Version (IMAVER)
- IMA Frame Size (RM)

Other parameters do not depend on ICP information for programmability. Therefore, they should be initialized before the start of the IFSM. Start the IFSM by setting each link in the group to “Group Assigned”:

- Set ILRCNTL[GA].

Management software must wait until each link in the group achieves IMA frame synchronization. For each link in the group that is “group assigned,” an IMA frame synchronization working (IFSW) event is generated. Having achieved frame synchronization, software can enable the group delay synchronization (GDS) mechanism; that is, finding the link with shortest delay and buffering accordingly through DCB.

- Configure the group order table with the ascending link order (round-robin distribution) to be used in reconstructing the ATM stream.
- Set the corresponding PHY bits in REF_LINK.
- Set IGRSTATE[GDSS] to 1 (one) to enable GDS.

After group delay synchronization is achieved, a GDS event is generated and ATM stream reconstruction can take place. For stream reconstruction to be performed by the processor, the user must switch all links and corresponding group (both directions) to active mode for receiving data cells. Set RX to active first to prevent the transmitter from generating a data stream when the opposite end is not ready:

- Set ILRCNTL[RXSC] to enable reception of data cells at the link level.
- Set IGRCNTL[RXSC] to enable reception of data cells at the group level.

- Set ILTCNTL[TXSC] to enable transmission of data cells at the link level.
- Set IGTCNTL[TXSC] to enable reception of data cells at the group level.

35.5.3.4.1 Initiator (Tx) Actions

Most of the actions required when a system initiates the establishment of a group with X links involve the exchange of ICP cells between the near end (initiator) and the far end (responder). Both ends must start with a group of X potential links configured to filler mode. One and only one of the links in the group must be designated as TRL.

- Set ILTCNTL[TXSC] to 0 (zero)—link is in filler mode.
- Set IGTCNTL[TXSC] to 0 (zero)—group is in filler mode.

The actions at both ends mirror each other. That is, the near end initiates the establishment of a group with X links by sending ICP cells to the FE and *vise versa*. The normal ICP state changes, driven by the state machine software, are (on a per-link basis):

- Not In Group
- Unusable
- Usable
- Active

Refer to Section 35.5.3.1, “Transmit ICP Cell Signaling” for details on how to modify and transmit an updated ICP cell. It is the responsibility of the GSM/LSM (group/Link State Machine software) to initialize the IMA ID (such as group ID) in the ICP cell template and link ID in the corresponding TX IMA Link Transmit Table Entry (ILTTE):

- Set ILTTE[ILID] = corresponding Link ID.
- Set IMA ID accordingly in the ICP Cell Template.

As an initiator, the NE (“end” is relative) must have established a group with X links provisioned.

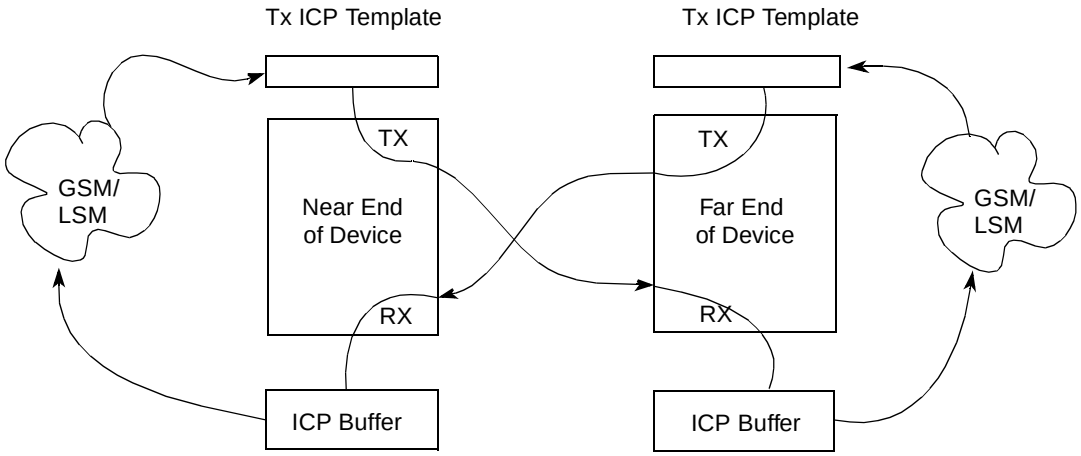


Figure 35-33. Near-End versus Far-End

35.5.3.4.2 Responder (Rx) Actions

The IMA GSM/LSM software receives ICP cells in the ICP buffer conveying the state of the FE group and links. After it determines that the required minimum number of links are available, it can proceed to enable the group delay synchronization (GDS) mechanism in which the all links in the corresponding group are analyzed to find the one with the shortest propagation delay. The amount of delay tolerated is programmable and is programmed by setting the beginning and ending addresses (size) of the delay compensation buffers (DCB). Note that the size of all DCBs must be the same and must be initialized before the GDS mechanism is activated. Activate GDS:

- Set IGRSTATE[GDSS] to enable GDS. After GDS is enabled, the user must not alter GDSS.

When group delay synchronization is achieved, a GDS event is generated and ATM stream reconstruction can take place. For the processor to perform ATM stream reconstruction, the user must switch all links in the group to active mode so that they can receive data cells, as specified earlier in this section. The initialization of the ATM TX and RX data structures required to generate and receive the ATM stream is out of the scope of this document, but it is documented in Chapter 30, “ATM Controller AAL0 and AAL5.”

35.5.3.5 Link Addition Procedure

Adding a link to an existing group requires changes to both the group and link table entries. Aside from the normal exchange of ICP cells by the state machines at the FE and NE, the steps described in this section should be followed. In general, a new link entry is appended to the existing list of link table entries and the required parameters initialized. This has no impact until the corresponding link is enabled through the PHYEN fields in the IMA root table.

The following sequence to achieve Rx link working state must be executed for each link independently of whether the link is added to a group either through LDS or GDS.

1. Reset all RX parameters in the new link table entry to zero.
2. Assign the corresponding group number for the new link: ILRCNTL[IGNUM] = x.
3. Assign the channel number for ICP cell reception: RICPCH = x.
4. Enable the desired interrupts: IRINTMSK = x.
5. Allow for the reception of ICP cells during the IFSM stage: ILRCNTL[MON_ICP] = 1.
6. Indicate that this link has not yet been assigned to a group: ILRCNTL[GA] = 0.
7. Configure this link/PHY as an IMA link by adding the link to the UTOPIA address compression table and updating the RXPHYEN_TSIZE.
8. Enable the corresponding link/PHY at the TC layer for example if using SAM as the TC layer set MTC_MODE[RXEN], or clear MTC_MODE[RAD] if the MTC is already in SYNC state but not passing cells up. If an external TC layer is used, set the corresponding enable bit for the external TC layer device.
9. Software receives and accepts ICP cell values (M, LID, and so on).
10. Program expected LID: ILID = x.
11. Program the expected ICP offset: LICPOS = x.
12. Initialize the DCB pointers accordingly: DCBEP, DCBSP, DCBFP. Note, it is recommended that the DCB be initialized to zero.

13. Configure link to loss of IMA frame state: ILRSTATE[FSES] = 2.
14. Start the IMA frame synchronization mechanism (IFSM) by assigning this link to the group: ILRCNTL[GA] = 1.
15. Software must now wait for the IFSW event.

The Rx steps for GDS, setting up a new group are as follows:

1. Formulate the new content of group order table RGRPORDER0 with the new link(s) included (see Section 35.4.4.2.4, “Receive Group Order Tables”), but do not flip GOTP (GOTP must remain 0).
2. Update RNUMLINKS. The stall threshold must be recalculated.This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS event). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 35.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
3. Start the delay compensation process for the link(s) by setting the corresponding bit(s) in the group table REF_LINK parameter.
4. Start the GDS procedure as shown in Section 35.5.3.4, “Group Startup Procedure.”

   Software must now wait for the group delay synchronization process to complete. A group delay synchronized (GDS) event is generated as soon as this happens.
5. If the GDS completes successfully, it is now safe for the link(s) to receive data. See Section 35.5.3.4, “Group Startup Procedure,” on how to activate the group and its links.

The Rx steps for LDS, adding a link to an existing group proceed as follows. This procedure can be invoked only if a GDS has successfully completed for the group.

1. Now that we have a “frame” synchronized link, we can proceed to allow the link to be delay synchronized. Assuming ILRCNTL[ADD_NEW] = ILRCNTL[ADD_NEW] indicate that this is a new link by inverting the current “add-new” bit value: ILRCNTL[ADD_NEW] = x. Basically ensure that ILRCNTL[ADD_NEW] is not equal to ILRSTATE[ADD_NEW_M].
2. Formulate a new group order table with the new link included (see Section 35.4.4.2.4, “Receive Group Order Tables”).
3. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
4. Update RNUMLINKS. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS event). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 35.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
5. Start the delay compensation process for this link (in IMA Root Table) by setting the corresponding bit in REF_LINK. See Table 35-3.
6. Software must now wait for the link delay synchronization process to complete. A link delay synchronized (LDS) event is generated by the QUICC Engine module as soon as this happens.
7. It is now safe for the link to receive data; set the link to active mode: ILRCNTL[RXSC] = 1.

To configure the Tx parameters, perform the following steps:

1. Reset all TX parameters in the new link table entry to zero.
2. Assign corresponding group number for the new link: ILTCNTL[IGNUM] = x.
3. Enable desired interrupts: ITINTMSK = x.
4. Software formats content of ICP template accordingly (see section “Transmit ICP Cell Signaling”).
5. Program the Link’s ID (LID) in the IMA Link Transmit Table Entry (ILTTE): ILID = x.
6. Program the ICP offset (ILTTE): LICPOS = x.
7. Initialize the Transmit Queue pointers accordingly: ITQSP, ITQEP, ITQFP, and ITQXP.
8. Construct the new TX Group Order Table (includes the added/new link).
9. Point to new TX Group Order Table in the corresponding IMA Group Transmit Table Entry (IGTTE): TGRPORDER = New Table Offset.
10. Configure this link/PHY as an IMA link by adding an entry to the UTOPIA compression and then updating TXPHYEN_TSIZE parameter to reflect the new table size.
11. Enable the corresponding link/PHY by enabling the TC layer. Up to this stage the TC layer should be disabled and should not request any cells from the IMA or ATM layers. For example if the TC layer for this link is polled on the UTOPIA bus it should not respond to the TxClav (requesting a cell) before this stage is complete. If the SAM is the TC for the link being added software should set the MTC_MODE[TXEN] at this stage and not before.
12. Software must now wait for the corresponding FE link to go to active state. (See Section 35.4.4.1.4, “ICP Cell Templates.”) The link can be configured to active mode to send data cells. Prior to this point, only filler and ICP cells were transmitted. ILTCNTL[TXSC] = 1.
13. Increment the number of links currently in the group (IGTTE): TNUMLINKS += 1.
14. If this link is the TRL set its ILTCNTL[TRL]. This bit must not be set until the previous steps have completed.

35.5.3.6 Link Removal Procedure

Removing a link from an existing group requires changes to both the group and link table entries. Aside from the normal exchange of ICP cells by the state machines at the FE and NE, the following steps should be followed. In general, a link entry is removed from the existing list of link table entries and the required parameters initialized.

35.5.3.6.1 Rx Link Removal

1. Formulate new group order table with the “dropped” link excluded (see Section 35.4.4.2.4, “Receive Group Order Tables”).
2. Update RNUMLINKS. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link; see LS event). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See section “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
3. Inhibit storing of cells in DCB for “dropped” link (in IMA Root Table): REF_LINK &= ~x (such as, clear the corresponding link bit in the REF_LINK entry).

4. Indicate that the link should be dropped: ILRCNTL[RXSC] = 2.
5. Software should wait (poll) for removal of the link from the DCB routine. The corresponding bit in the group table LINK_DCB entry is cleared, so no more cells are stored in the DCB. For example, while ((LINK_DCB & REF_LINK_BITMASK) != 0)/* wait */;. If this wait does not succeed within RNUMLINKS*STALL_THR cell times on the physical interface, we must assume that all links in the group are down. See Section 35.5.3.6.2, “Rx Steps No RxClav For All Member Links In A Group. The value of RNUMLINKS used must be the value before decrementing in step 2.
6. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
7. Indicate that this link is no longer assigned to a group: ILRCNTL[GA] = 0.
8. Inhibit reception of cells over the dropped link by disabling the TC layer such that it does not pass any more cells to the IMA layer. Disable the TC layer, for example, if using the SAM clear MTC_MODE[RXEN]. If the TC layer is connected on the UTOPIA bus, disable it by ensuring that it passes no more cells onto the UTOPIA bus for this link. If the application requires that the TC layer remains cell delineated (no LOCD) at this step then software should adhere to the following steps:
    a) Leave MTC_MODE[RXEN] = 1
    b) Set MTC_MODE[RAD]
9. Wait a period of time that is equal to 2 cells time on the physical interface (for example, 2 cells duration on an E1 interface).
10. Software should wait (poll) for the processor to use the new group order table. This simply ensures that it is safe to modify/reuse the dropped link parameters because the processor is no longer using the dropped link’s data structures. Ensure that the group order pointer points to the new group order table (at this point, no more cells are extracted out of the dropped link’s DCB). For example, use while (dcblink != new_pointer). DCBLINK is an entry in the IGRTE. If this wait does not succeed within RNUMLINKS*STALL_THR cell times on the physical interface, we must assume that all links in the group are down. See Section 35.5.3.6.2, “Rx Steps No RxClav For All Member Links In A Group,” in that case. The value of RNUMLINKS used must be the value before decrementing in step 2.

35.5.3.6.2 Rx Steps No RxClav For All Member Links In A Group

Note that if only one link is used in a group the software must monitor the TC layer to detect that this link has stalled. If all links in the group go down simultaneously (for example, due to removal of cables), this scenario should be treated as the last link stall case and the TC layer indications must also be used to determine that the links should be removed. In general, if the TC layer no longer passes cells to the IMA layer (for example, no RxClav assertions for all links in the group), the removal procedure in Section 35.5.3.6.1, “Rx Link Removal,” will not be successful. Therefore, if all links in a group are no longer passing cells, follow the procedure in Section 35.5.3.6.2, “Rx Steps No RxClav For All Member Links In A Group.” This procedure should also be used if the group is to be torn down during GDS.

1. Inhibit reception of cells over the dropped link(s) by disabling the TC layer(s) so that they do not pass any more cells to the IMA layer. Disable the TC layer(s), for example, if the SAM clear MTC_MODE[RXEN] is used for each link. If the TC layer(s) are connected on the UTOPIA bus, disable each one to ensure that they pass no more cells onto the UTOPIA bus.

2. Wait a period of time that is equal to 2 cells time on the physical interface (for example, 2 cells duration on an E1 interface)
3. It is now safe to configure the group and link tables to their default values.
4. If all or some of the links are to be used again, follow the group initialization procedure (see Section 35.5.3.4, “Group Startup Procedure”).

35.5.3.6.3 Tx Parameters For Non-TRL Link

1. Inhibit transmission of cells over the dropped link by disabling requests from the link for cells. The TC layer (PHY number for this link) should not request any more cells on the UTOPIA bus. If SAM is used, clear each TC layer MTC_MODE[TXEN].
2. Formulate the new group order table with the dropped link excluded (see Section 35.4.4.1.3, “Transmit Group Order Table”).
3. Point to new TX group order table in the corresponding IMA group transmit table entry (IGTTE): TGRPORDER = New Table Offset.
4. Decrement the number of links in the group (IGTTE): TNUMLINKS -= 1.
5. Wait a period of time that is equal to 2 cells time on the physical interface (for example, 2 cells duration on an E1 interface).

35.5.3.6.4 Tx Parameters TRL Link

The TRL link removal requires all other links in the group to be removed. because the TRL controls the filling of the transmit queues of all links in the group. After the TRL is removed, all other links queues are no longer passed cells. This procedure should also be performed when the last link is removed from the group because by definition the last link must be the TRL.

1. Inhibit transmission of cells over all links in the group by disabling requests from these links for more cells. The TC layers (PHY numbers for each link) should not request any more cells on the UTOPIA bus. If SAM is used, clear each TC layers MTC_MODE[TXEN].
2. Remove the TRL by clearing ILTCNTL[TRL].
3. For all member links, execute the TX link removal procedure as described in Section 33.5.4.5.3, “TX Link Removal.

35.5.3.7 Link Receive Deactivation Procedure

The following procedure assumes that the link was part of the IMA group during the group delay synchronization procedure and that an IFSW (IMA Frame Synchronization Working) event has already been received for the link.

1. Update RNUMLINKS. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS event). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 35.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
2. Inhibit storing of cells in DCB for “dropped” link (in IMA Root Table): REF_LINK &= ~x (such as, clear the corresponding link bit in the REF_LINK entry).
3. Indicate that the link should be dropped: ILRCNTL[RXSC] = 2.

4. Software should wait (poll) until the link is removed from the DCB routine. The corresponding bit in the group table LINK_DCB entry is cleared so that no more cells are stored in the DCB). For example, use while ((LINK_DCB & REF_LINK_BITMASK) != 0) /* wait */;. If this wait does not succeed within RNUMLINKS*STALL_THR cell times on the physical interface, we must assume that all links in the group are down. See Section 35.5.3.6.2, “Rx Steps No RxClav For All Member Links In A Group.” The value of RNUMLINKS used must be the value before decrementing in step 1.
5. Formulate new group order table with the dropped link excluded (see Section 35.4.4.2.4, “Receive Group Order Tables”).
6. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
7. Set the link to filler mode ILRCNTL[RXSC] = 0.
8. Initialize the DCB pointers accordingly: DCBSP = DCBFP, DCBRP = Null.
9. Initialize link DCB in external memory to zero.
10. If this was the last link in group (such as involved in passing cells from the DCB to the ATM microcode) software must also clear IGRSTATE[GDSS].

35.5.3.8 Link Receive Reactivation Procedure

The following procedure assumes that the link is part of the IMA group during the group delay synchronization procedure and that an IMA frame synchronization working (IFSW) event has already been received for the link.

1. Indicate that this is a new link (GDS/reconstruction function) by inverting the current “add_new” bit value: ILRCNTL[ADD_NEW] = x.
2. Formulate the new group order table with the new link included (see Section 35.4.4.2.4, “Receive Group Order Tables”).
3. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
4. Increment RNUMLINKS in the group receive table.
5. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS event). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 35.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
6. Start the delay compensation process for this link (in IMA Root Table) by setting the corresponding bit in REF_LINK. See Table 35-3.
7. Software must now wait for the link delay synchronization process to complete. A LDS (Link Delay Synchronized) event will be generated by the processor as soon as this happens.
8. It is now safe for the link to receive data, set link to “active” mode: ILRCNTL[RXSC] = 01.

35.5.3.9 Transmit Event Response Procedures

The following TX events can occur in IMA mode. It is recommended that all events be handled through an exception/interrupt service routine (ISR) because the response time inherent with interrupt-driven events should diminish the negative impact/propagation of such events:

- Transmit queue underrun (TQU)—Indicates that a transmit queue was emptied. This implies that the offending link (PHY) is requesting cells at a faster rate relative to the TRL. If the TRL is out of spec, then multiple links report TQUs because all links in the group are faster relative to the TRL. The offending link should be removed (see Section 35.5.3.6, “Link Removal Procedure”). Check the following: PHY, TX Queue depth (start and end pointers should not be the same); TNUMLINKS is equal (not less or greater) than the number of active links.
- Transmit queue overflow (TQO)—Indicates that a transmit queue was not ready to receive a cell (full). This implies that the offending link (PHY) is requesting cells at a slower rate relative to the TRL. If the TRL is out of spec, then multiple links report TQOs because all links in the group are slower relative to the TRL. The offending link should be removed (see Section 35.5.3.6, “Link Removal Procedure”). Check the following: PHY, TX Queue depth (depth should be the same as other queues); TNUMLINKS is equal (not less or greater) than the number of active links.

35.5.3.10 Receive Event Response Procedures

The following RX events may take place when operating in IMA mode. It is recommended that all events be handled via an “exception/interrupt service routine” (ISR) as the response time inherent with interrupt driven events should diminish the negative impact/propagation of such events:

- Link stalled (LS)—The link’s DCB has emptied, either because the PHY is no longer functioning or it is relatively slower than the other links. The offending link should be removed (see Section 35.5.3.6, “Link Removal Procedure”). If only one link is used in a group, the software must monitor the TC layer to detect that this link has stalled. No LS event is reported in the IMA interrupt queue for this case. To restart an IMA group properly after last/only link has recovered, the user must follow the group startup procedure.
- DCBO (DCB Overflow)—The delay compensation buffer has no more space. The source of the problem can be varied:
  - — the offending PHY is sending at a faster rate (out of spec.)
  - — the propagation delay of one of the other links in the group is greater than anticipated (need to make DCB larger)
  - — the size of the offending link was programmed incorrectly, that is, smaller (the size of all DCBs in the group should be the same).

  The offending link should be removed or deactivated (see Section 35.5.3.6, “Link Removal Procedure” or Section 35.5.3.7, “Link Receive Deactivation Procedure”). This event corresponds to the link out of delay synchronization defect (LODS) and should be reported to the FE (through the ICP cell, RxDefect = LODS). The processor continues to report this event if the condition persists (unless the event is masked).
- Link delay synchronized (LDS)—The new/added link to an existing group has achieved synchronization (link delay). The link can receive data at this point, (see Section 35.5.3.5, “Link Addition Procedure”).
- Group delay synchronized (GDS)—Group delay synchronized achieved or not achieved. In some cases, the GDS cannot complete. This can happen if a link experiences problems during the GDS process, for example, losing SYNC state for the IFSM. If this occurs, it is not possible to determine the link’s true differential delay with respect to other member links of the group. To determine

whether the GDS process completed successfully, the ucode sets IGRSTATE[GDSS] to either of the following values after the GDS interrupt is generated:

— IGRSTATE[GDSS] = 00 – GDS process failed, a link lost IFSM SYNC during GDS process

— IGRSTATE[GDSS] = 11 – GDS process complete

Software can then read IGRSTATE[GDSS] and use this status information before deciding whether to change the links to the active state or to try to resynchronize again at the group level. In addition to the preceding status information, the ILRSTATE[ADD_NEW_M] bit of the link that caused the failure of the GDS process is flipped so that it is logically inverted with respect to the ILRCNTL[ADD_NEW] bit. If the GDS fails software can restart a new GDS with the link that caused the failure excluded. Software must reset the DCB pointers for all of the links to their default values. If the excluded link subsequently reaches the WORKING state it can be added to the group again using the LDS procedure see Section 35.5.3.5, “Link Addition Procedure,” for more information. Note that a link losing SYNC during the GDS process can cause DCBO interrupt for other links in the group. If this situation occurs the GDS process should be restarted as described.

- Link (IMA) frame synchronization defect (IFSD)—The link has lost synchronization (for example, unexpected IFSN value for a number (GAMMA + 2) of consecutive frames). The interrupt will be issued every (GAMMA + 2) IMA frames whilst the IMA Error/Maintenance State Machine is in the DEFECT state. If this condition persists, the software can choose to remove the link after a certain threshold of consecutive IFSD interrupts are encountered. The threshold that would trigger the removal of the link is system-specific. Alternatively the software can leave the link in the group assigned state and wait for the link to recover, the recovery will be flagged through an IFSW interrupt. The IFSD event corresponds to the “Loss of IMA Frame” (LIF) state and the software should react to this by informing the FE of the problem (through the ICP cell, RxDefect = LIF). The processor maintains counters (if enabled) to track the overall quality of a particular link: see OIF (Out of IMA Frame) and ICPVIOL (ICP Violation) in Section 35.4.5.3, “IMA Link Receive Statistics Table.” The software can use one of the processor timers as a time stamp when handling IFSD events and check if the persistence time threshold has been crossed when subsequent events occur. The LIF state time stamp should be reset when the IFSW event is reported.

NOTE

If the periodic IFSD interrupts handling is not required by the software, upon the first instance of IFSD for this link the software can mask the interrupt. Then the software can wait for the IFSW interrupt which signals IFSM recovery and the working state for this link. When processing the IFSW interrupt it is recommended that the IFSD is unmasked.

- Link (IMA) frame synchronization working (IFSW)—The link has achieved frame synchronization. This event is reported when a link has been assigned to a group (startup or link addition) or it was previously assigned and working, but had a defect (see IFSD event). If going from defect to working, the software should update the RxDefect field of the ICP. Again, a time stamp (timer) can be used to measure the persistence time.
- DCB synchronization lost (DSL)—A link in a group with IGRSTATE[GDSS] = 11 loses synchronization and enters HUNT state at the IFSM. Because the link has lost synchronization, its differential delay with respect to other member links in the group must recalculated via the LDS.

When this interrupt occurs, software should remove or deactivate the link because the QUICC Engine module does not automatically perform the LASR procedure. Note that when the interrupt is generated the ILRSTATE[DL] bit is set. If the link is to be added to the group again, software can perform the link removal or deactivation procedure described in Section 35.5.3.6, "Link Removal Procedure or Section 35.5.3.7, "Link Receive Deactivation Procedure." The link can then be re-added to the group using the link addition procedure see Section 35.5.3.5, "Link Addition Procedure," for a removed link or Section 35.5.3.8, "Link Receive Reactivation Procedure," for a deactivated link. For fast recovery from the DSL event the software should perform the following tasks:

— Link deactivation; see Section 35.5.3.7, "Link Receive Deactivation Procedure."
— Poll ILRSTATE[FSES] checking for WORKING state, ILRSTATE[FSES] = 0b00. The poll of ILRSTATE[FSES] must last at least (GAMMA +1) IMA frames.
— If WORKING state is reached then perform reactivation, see Section 35.5.3.8, "Link Receive Reactivation Procedure."
— Else if ILRSTATE[FSES] = 0b1x then the DEFECT state has been reached and an IFSD event will be generated for this link.

The relationship of the DSL interrupt versus the IFSD interrupt is depicted in Figure 35-34. The DSL interrupt is issued when the IFSM reaches the HUNT state. The SYNC to HUNT transition also creates the ANOMALY state for the IMA Error and Maintenance State Diagram. The link can recover from the ANOMALY state and return to the WORKING state before an IFSD. The DSL is issued only for links that have achieved GDS or LDS or for links that have started but not completed the LDS process.

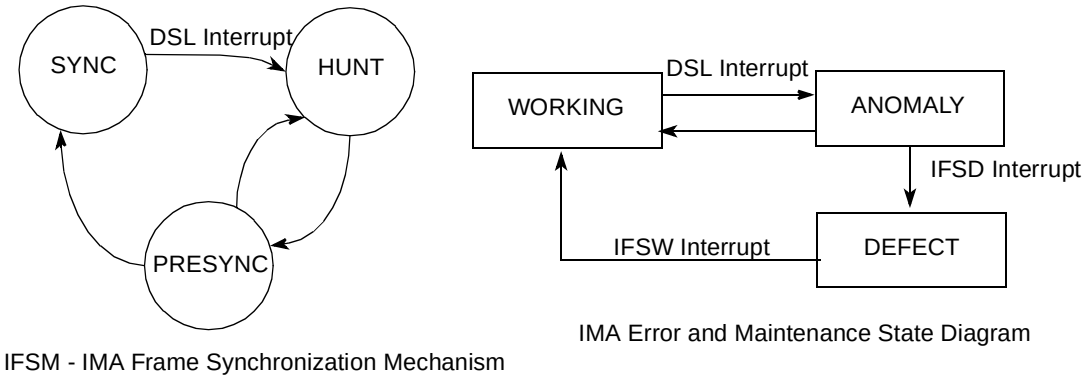


Figure 35-34. Receive Event Generation For ATM Forum IMA State Machines

35.5.3.11 Test Pattern Procedure

Test patterns verify the "connectivity" of a link in a particular group. Through an ICP cell, the NE requests that a test pattern be looped back to the FE over all links currently in the group. For this test, the FE looks only at ICP cells received over the link being tested (LID = x). Upon receiving an ICP cell, the FE echoes the pattern back to the NE over all the links in the group. The major task of initiating and verifying the pattern on all the links in the group falls on the software. The processor simply transmits and receives the

modified/changed ICP cells. If the software detects that the FE has echoed the pattern, connectivity is verified.

35.5.3.11.1 As Initiator (NE)

Alter the (unused) ICP cell template to the “Test Link” command to the FE:

1. Tx Test Control = Set to “Active” and select LID (link) to be tested.
2. Tx Pattern = X (0–255).
3. Increment SCCI.
4. Make this the active ICP template (see Section 35.5.3.1, “Transmit ICP Cell Signaling”).
5. Repeat steps 1–4 until the expected pattern is received in any of the incoming ICP cells (alternate ICP templates must be used).

After the request is sent to perform the test (pattern) to the FE, the NE software must wait for the TX test pattern to be echoed back/received in the RX Pattern field of the ICP cell (any of the active links in the group can be monitored for the RX Pattern). After verifying connectivity, the test can be deactivated (set Tx Test Control = “Inactive” and repeat steps 3 and 4).

35.5.3.11.2 As Responder (FE)

When the FE receives a request to perform the pattern test on any active links, it must monitor the selected link (see Tx LID of Tx Test Control) for reception of an ICP cell. It must also monitor the other links’ ICP cells and ensure that all links in the group have a valid Test Link Command field. After verifying that all links in the group have a valid Test Link Command field in their ICP cells, software should copy the Tx Pattern from the test link to the Rx Pattern field of the ICP Cell and transmit the ICP cell. Alter the ICP cell:

1. Rx Pattern = Tx Pattern (of received ICP Cell).
2. Increment SCCI
3. Make this the active ICP template (see Section 35.5.3.1, “Transmit ICP Cell Signaling”).
4. Repeat steps 1-3 until the test is inactive.

To comply with this requirement all links in the group must have MON_ICP bit set:

1. Monitor link for changes in SCCI: ILRCNTL[MON_ICP] = 1.

The IMA microcode passes to the ICP cell queue all received ICP cells with a Link Test Command field displaying the active state. This is true regardless of the SCCI value received. The SCCI field is ignored in this case in order to comply with the requirement that all links in the group must detect a Link Test Command field in the active state before passing the test link’s Tx Test Pattern field to the transmitter’s Rx Test Pattern field.

35.5.3.12 End-to-End Channel Signaling Procedure

35.5.3.12.1 Transmit

Software can use the end-to-end channel signaling field to signal to the far end. Because the end-to-end channel is not covered by the SCCI field and there are no standard requirements for the minimum number

of frames between changes of the end-to-end channel field, it is not necessary to follow the procedure for ICP cell signaling to do this; instead, the end-to-end channel field of the ICP template currently in use can be directly updated. However, if a minimum number of frames between changes is desired, then the procedure for ICP cell signaling can be used, skipping the step in which the SCCI is updated.

35.5.3.12.2 Receive

No special facility is included for reception of the end-to-end channel. The end-to-end channel field is part of the received ICP cells, so its information can be read by software from those cells. However, ICP cells are only received when the SCCI field changes, so changes in the end-to-end channel will not be received until the SCCI field also changes (when there is a change in the status and/or control of the far end).

Chapter 36
Universal Serial Bus Controller

The universal serial bus (USB) controller provides communication with other devices through a USB connection.This chapter describes the QUICC Engine USB controller, including basic operation, the parameter RAM, and registers.

36.1 Overview

The USB controller is an industry-standard extension to the PC architecture. The QUICC Engine USB controller 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 *USB Specification Revision 2.0*. This specification can be downloaded from http://www.usb.org.

The QUICC Engine 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).

36.1.1 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).
- Flexible data buffers with multiple buffers per frame
- Automatic transmission of SOF (Start-of-Frame) tokens
- Supports local loopback mode for diagnostics (12 Mbps only)

36.2 Host Controller Limitations

The following tasks are not supported by the hardware and must be implemented in software:

- Scheduling the various transfers within and between frames
- Retransmission after an error and error recovery

Additionally, when the packet-level interface described in Section 36.4.1.1, “Packet-Level Interface” is in use, the tokens must be prepared by the software. Because the QUICC Engine USB host controller does not integrate the root hub, an external hub is required when more than one device is connected to the host. Also, note that the host controller programming model does not conform to the open host controller interface (OHCI) or universal host controller interface (UHCI) standards in which software drivers are hardware-independent.

36.2.1 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.

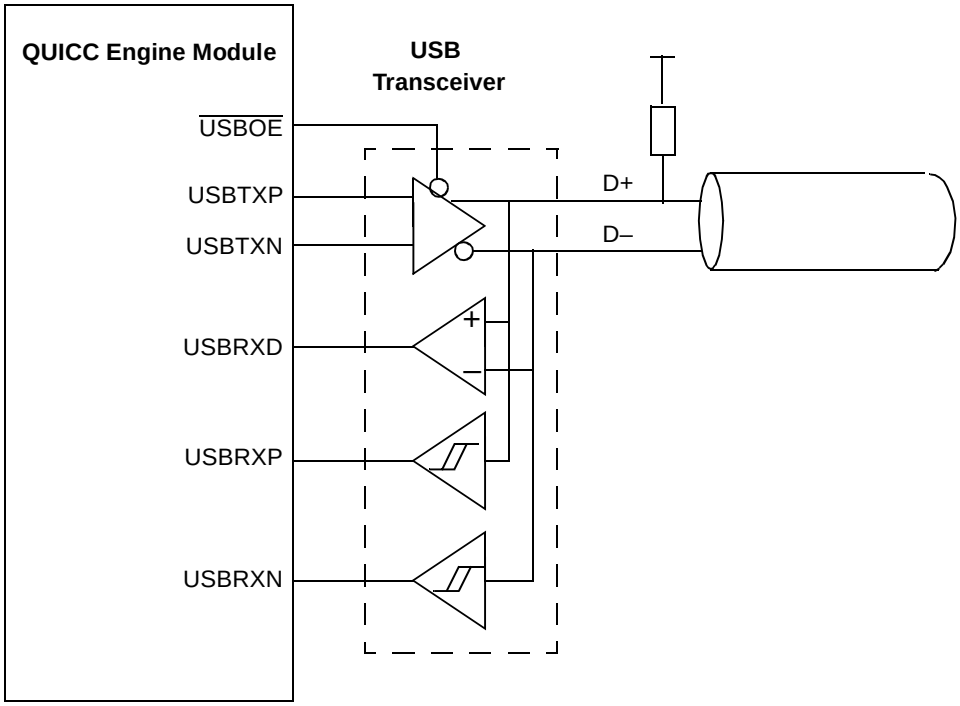


Figure 36-1. USB Interface

The DPLL circuitry uses the reference clock for the USB controller (USBCLK) to recover the bit rate clock. The source for USBCLK is selected by GMXGCR[USBCS]; see Section 20.5.1, "CMX General Clock Route Register (CMXGCR)." The QUICC Engine module 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 36-1. 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 36-1. USB Pin 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 36-1. USB Pin Functions (continued)

<table>
<tr><th>Signal</th><th>I/O</th><th>Function</th></tr>
<tr><td>USBRXD</td><td>I</td><td>Receive data. Input to the USB receiver from the differential line receiver.</td></tr>
<tr><td>USBRXP,
USBRXN</td><td>I</td><td>Gated version of D+ and D–. Used to detect single-ended zeros and the interconnect speed.
<table>
<tr><th>RP</th><th>RN</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>Single-ended 0</td></tr>
<tr><td>1</td><td>0</td><td>Full speed</td></tr>
<tr><td>0</td><td>1</td><td>Low speed</td></tr>
<tr><td>1</td><td>1</td><td>—</td></tr>
</table>
</td></tr>
</table>

36.3 USB Function Description

As shown in Figure 36-2, 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.

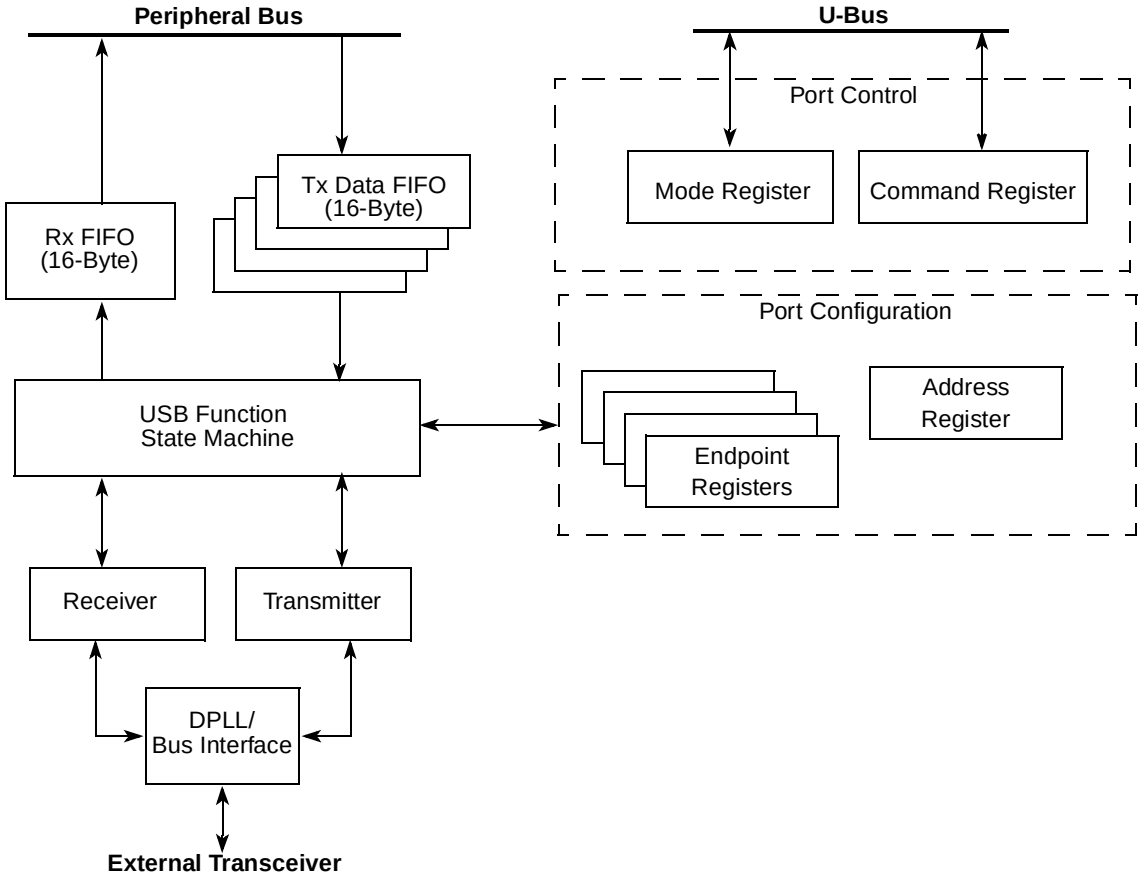


Figure 36-2. USB Function Block Diagram

36.3.1 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 36.4.7.2, “USB Slave Address Register (USADR)”) should be programmed with the assigned address. The USB function 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 36.4.7.3, “USB Endpoint Registers (USEP0–USEP3)”).

NOTE

It is mandatory that endpoint 0 be configured as a control transfer type. The USB system software uses this endpoint as a control pipe. Additional control pipes can be provided by other endpoints.

After it is enabled, the USB function controller looks for valid token packets. Figure 36-3 and Table 36-2 describe the behavior of the USB controller for each token. The USB function controller ignores tokens that are not valid (that is, PID check fails or CRC check fails or packet length is not 3 bytes).

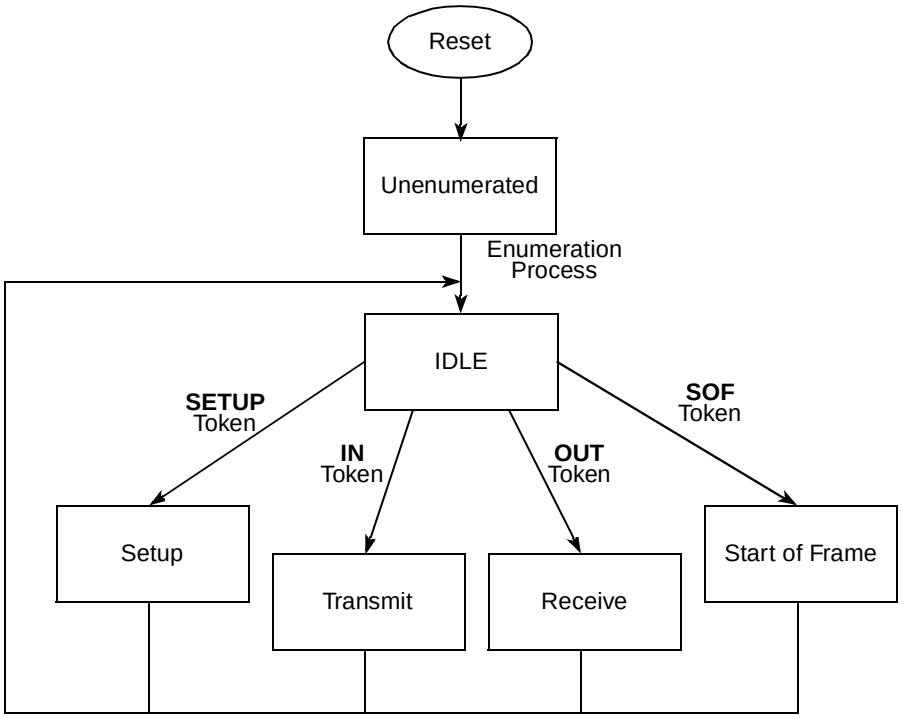


Figure 36-3. USB Controller Operating Modes

Table 36-2. 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. When the token is received and the FIFO is loaded with the last data byte or with at least four bytes, transmission begins. The four-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] is set to respond with STALL, a STALL handshake is returned, as shown in the following table. 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 (data discarded)</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 (data discarded)</td></tr><tr><td>11 (STALL)</td><td>—</td><td>STALL</td></tr></table></td></tr>
</table>

Table 36-2. 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.

36.4 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 (see Figure 36-4). The USB controller consists of transmitter and receiver sections, a host control unit, and a function control unit for testing purposes. The USB transmitter contains four independent FIFOs, each containing 16 bytes. Endpoint 0 is dedicated to host transactions; endpoints 1–3 are for function transactions in test mode. There is a dedicated FIFO for each of the four supported endpoints; the endpoint 0 FIFO is for host transactions. The USB receiver has a single 16-byte FIFO.

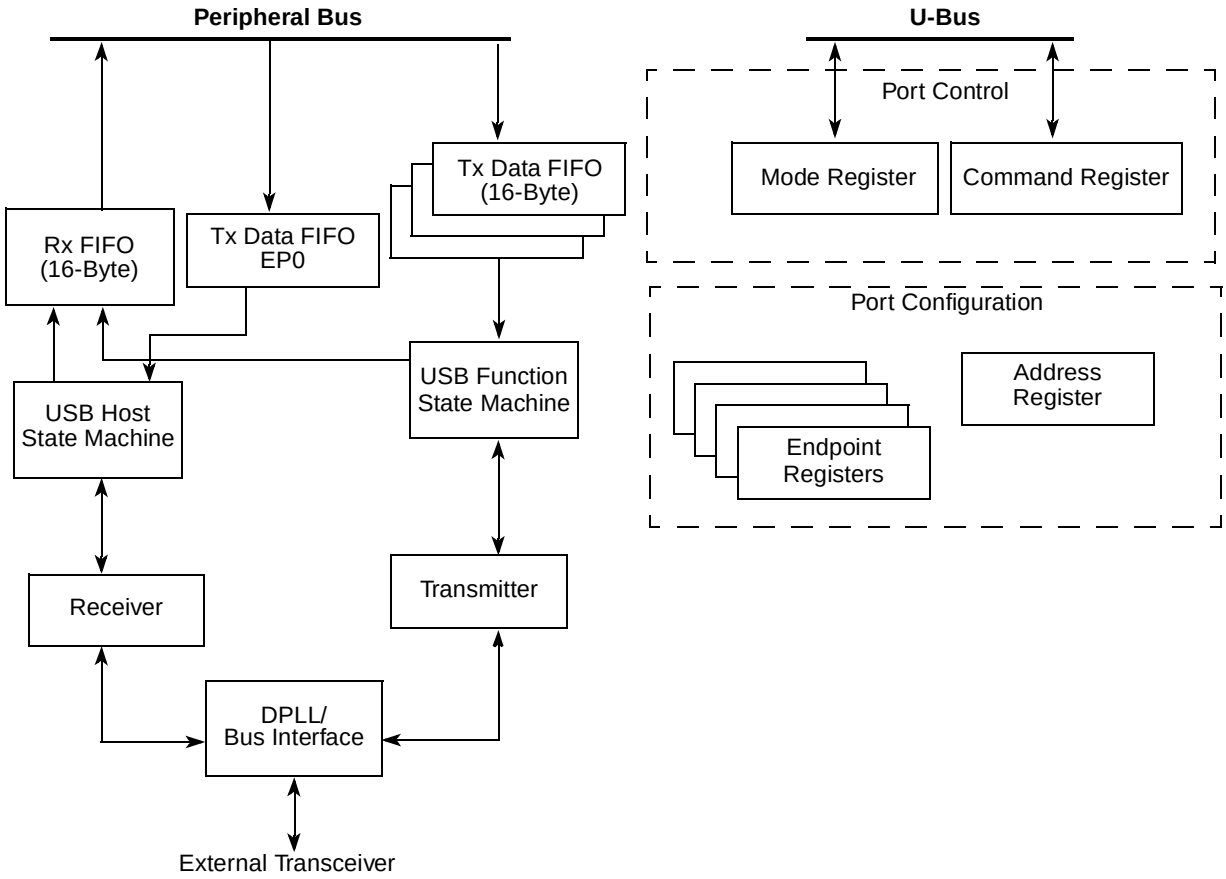


Figure 36-4. USB Controller Block Diagram

36.4.1 USB Host Controller Transmit/Receive

The USB host controller initiates all USB transactions in the system. After reset, the USMOD[HOST] bit should be set to enable host operation (refer to Section 36.4.7.1, “USB Mode Register (USMOD)”). USEP1 should be programmed for host operation as described in Section 36.4.7.3, “USB Endpoint Registers (USEP0–USEP3).”

After it is enabled by setting the USMOD[EN] bit, the USB host controller waits for a packet in its transmit FIFO. When the FIFO contains data for transmission, the host transaction begins. Figure 36-3 and Table 36-2 describe the behavior of the USB host controller for each transaction. Low-speed transactions start with a preamble that is generated by the USB host controller state machine when the TxBD[LSP] bit is set. When USMOD[TEST] is programmed, both the host state machine and function state machine are active. Endpoints 2–4 receive/transmit data according to tokens received from host. For the programming model and a functional description, see Section 36.4.7, “USB Function Programming Model.”

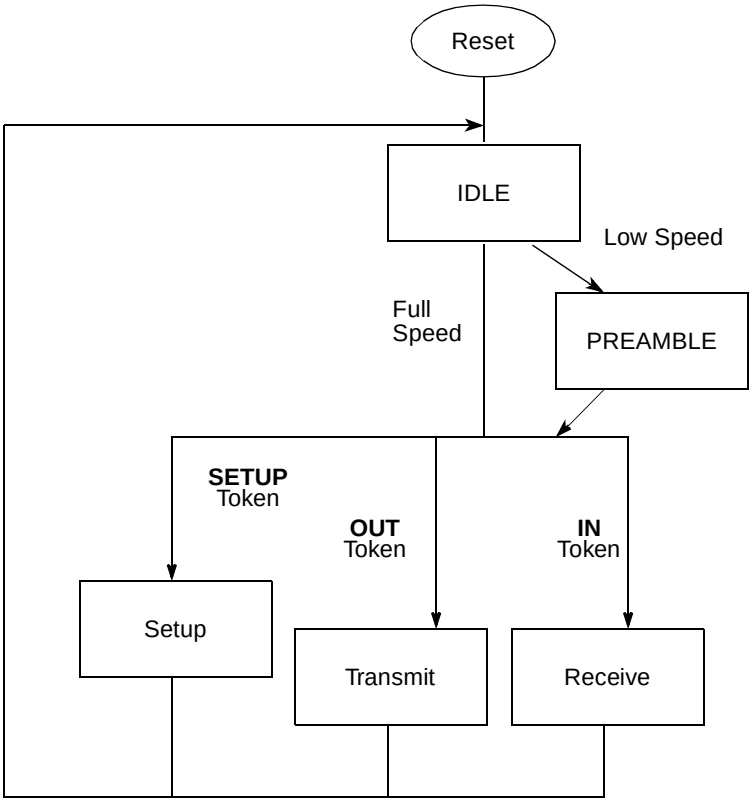


Figure 36-5. USB Controller Operating Modes

36.4.1.1 Packet-Level Interface

If USEP0[RTE] is cleared, 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 36.5.3, "USB Transmit Buffer Descriptor (TxBD) for Host." Each receive packet is stored in a buffer referenced by a RxBD as described in Section 36.5.1, "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.

36.4.1.2 Transaction-Level Interface

If USEP0[RTE] is set, the USB host controller uses a transaction-level interface to communicate with the user. Each transaction uses one TrBD as described in Section 36.5.4, "USB Transaction Buffer Descriptor (TrBD) for Host." The USB host controller generates the token based on TrBD[TOK]. 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 that is used for the receive data packet.

Table 36-3. 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>
</table>

USB Out Transaction

Token	Data	Handshake Received by Host	Indication on TxBD/TrBD
OUT	Sent by host	None	TO
		ACK	None
		NAK	NAK
		STALL	STALL

Table 36-3. 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 is 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 nonoctet 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 nonoctet 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 colspan="3">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 and cannot be answered with NAK or STALL, therefore, the host expects either an ACK or no handshake at all.</td></tr>
<tr><td>Start of Frame (SOF)</td><td colspan="2">SOF is generated every 1 ms. The timing must be exact and is controlled by an internal timer. From the host state machine point of view it is a packet to transmit, placed in its FIFO, transmitted as is.</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 host controller generates a full-speed PRE token before sending a packet to a low-speed peripheral.</td></tr>
</table>

36.4.2 SOF Transmission for USB Host Controller

The mechanism used by the USB host controller to support the automatic transmission of SOF tokens is enabled by setting USMOD[SFTE]. SOF packets should be transmitted every 1 ms. Because the time interval between two SOF packets must be more precise than software can accomplish, a hardware timer is used to trigger the generation and transmission of SOF tokens. When the timer expires, the frame number (stored in the USB Frame Number register) is incremented, and an SOF token is generated and loaded to the host endpoint. When the SOF token is loaded to the FIFO, it is transmitted like any other packet.

The application software should guarantee that the USB host completes 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 and frame number can be read at any time to synchronize the software with the USB frames. See Section 36.4.7.8, “USB Start of Frame Timer (USSFT)” and Section 36.4.7.9, “USB Frame Number (USFRN).” Failure to ensure that the FIFO is empty at the end of the frame period may result in the SOF packet not being transmitted. This situation is reported by setting the MSF bit of the USB Event Register.

36.4.3 USB Function and Host Parameter RAM Memory Map

The USB controller parameter RAM area, shown in Table 36-4, 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 36-4. 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 36-6. The map of the endpoint parameter block is shown in Table 36-5 **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	**EP3PTR**	Half word	
USB Base + 08	RSTATE	Word	Receive internal state. Reserved for QUICC Engine module use only. Should be cleared before the USB controller is enabled.
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.
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 QUICC Engine module use only.
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 QUICC Engine module.

After they are 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.

36.4.4 Endpoint Parameters Block Pointer (EP*n*PTR)

EP*n*PTR are DPRAM indices to an endpoint parameter block, which 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 36-6.

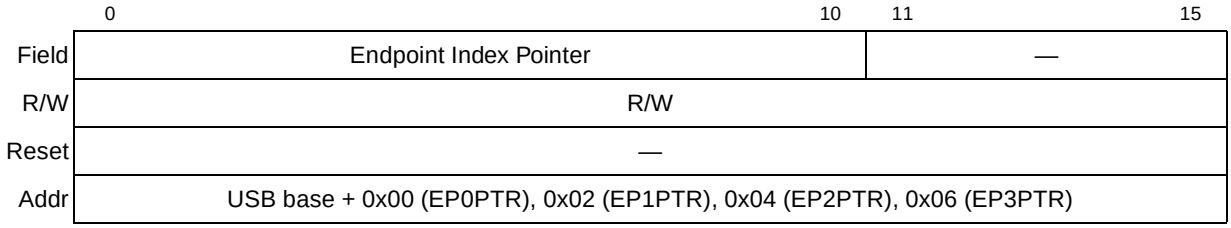


Figure 36-6. Endpoint Pointer Registers (EP*n*PTR)

The map of the endpoint parameter block is shown in Table 36-5.

Table 36-5. 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. When the transaction-level interface is used in host mode, TBASE points to the TrBD ring, and RBASE is unused.
0x02	**TBASE**	16 bits	
0x04	**RBMR**	8 bits	Rx/Tx Bus Mode registers. They contain the transaction specification associated with DMA channel accesses to external memory. See Section 36.4.6, "USB Bus Mode Registers (RBMR and TBMR)."
0x05	**TBMR**	8 bits	
0x06	**MRBLR**	16 bits	Maximum receive buffer length. Defines the maximum number of bytes the QUICC Engine module writes to the USB receive buffer before moving to the next buffer. MRBLR must be divisible by 4. The QUICC Engine module 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]. When using the transaction-level interface in host mode, this field is used by the QUICC Engine module and does not have to be initialized by the user.

Table 36-5. Endpoint Parameter Block (continued)

Offset[1]	Name[2]	Width	Description
0x08	**RBPTR**	16 bits	RxBD pointer. Points to the next BD the receiver transfers 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 QUICC Engine module 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. When the transaction-level interface operates in host mode, this field is unused.
0X0A	**TBPTR**	16 bits	TxBD pointer. Points to the next BD that the transmitter transfers data from when it is in an idle state or to the current BD during frame transmission. Software should initialize TBPTR after reset. When the end of BD table is reached, the QUICC Engine module. 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 QUICC Engine module 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 QUICC Engine module 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	Reserved	16 bits	—

Note:

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.

36.4.5 Frame Number (FRAME_N)

FRAME_IN is used for frame number updates in both function mode and host mode. In function mode, it is written by the USB controller; in host mode it is written by the application software and the USB controller. This entry is updated by the USB controller in function mode when an SOF (start of frame) token is received, including the SOF token it transmitted in host mode. The entry contains 11 bits that represent the frame number. An SOF interrupt is issued upon an update of this entry.

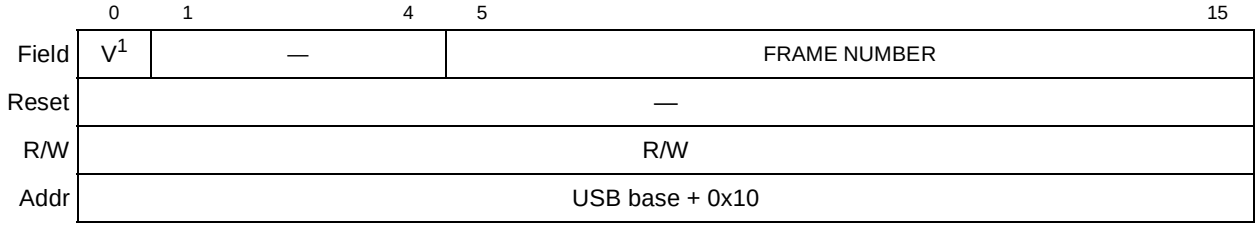


Figure 36-7. Frame Number (FRAME_N) in Function Mode—Updated by USB Controller

1 This bit is set if the SOF token is received error free.

Table 36-6 describes FRAME_N fields in function mode.

Table 36-6. 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.

In host mode, this entry must be updated by the application software between the transmission of one SOF (start of frame) token and the next. See Section 36.4.1.2, “Transaction-Level Interface,” for details. The software should prepare the frame number and the CRC and place it in FRAME_N field.

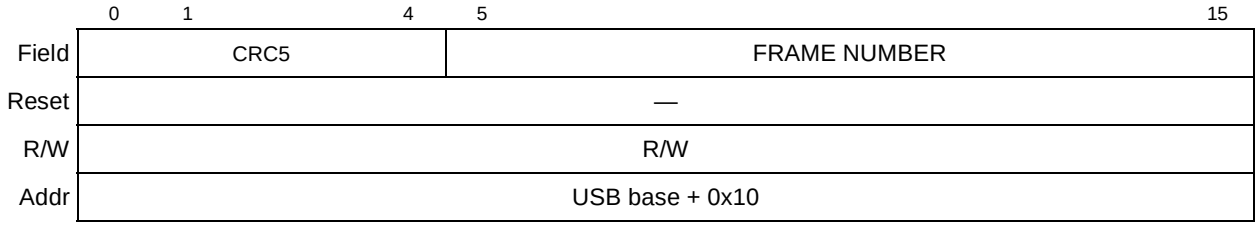


Figure 36-8. Frame Number (FRAME_N) in Host Mode—Updated by Application Software

Table 36-7 describes FRAME_N fields in host mode.

Table 36-7. 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.

36.4.6 USB Bus Mode Registers (RBMR and TBMR)

Figure 36-9 shows the fields in the receive/transmit bus mode registers.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	DTB	—

Figure 36-9. USB Bus Mode Registers (RBMR and TBMR)

Table 36-8 describes RBMR and TBMR fields.

Table 36-8. RBMR and TBMR Fields

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 18.1.8.2, “Serial DMA Mode Register (SDMR).”
3–4	BO	Byte ordering The user configures 00, 01, 11 Reserved 10 Big-endian byte ordering. 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	CETM	QUICC Engine Transaction Mark The level of this bit is reflected on the main system bus on signal M1SCRD4 or M2SCRD4 or LSRCRD4 depending on the actual external bus where the transaction occurs. This is useful to mark on the bus the transactions initiated by the QUICC Engine module, for debug purposes.
6	DTB	Indicates on what bus the data is located 0 On the coherent system bus (CSB) 1 On the QUICC Engine secondary bus
7	—	Reserved, should be cleared.

36.4.7 USB Function Programming Model

The following sections describe USB controller registers.

36.4.7.1 USB Mode Register (USMOD)

USMOD, shown in Figure 36-10, 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							
Addr	0x006C0							

Figure 36-10. USB Mode Register (USMOD)

Table 36-9 describes USMOD fields.

Table 36-9. 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.
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 36.4.7.8, “USB Start of Frame Timer (USSFT)” and Section 36.4.2, “SOF Transmission for USB Host Controller,” for more information. 0 SOF timer is disabled 1 SOF timer is enabled
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 is 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:** Other bits of the USMOD should not be modified by the user while EN is set.

36.4.7.2 USB Slave Address Register (USADR)

USADR holds the address for this USB port when operating as function.

	0	1 – 7
Field	—	SADx
Reset	0000_0000	
R/W	R/W	
Addr	0x006C1	

Figure 36-11. USB Slave Address Register (USADD)

Table 36-10 describes the USADR fields.

Table 36-10. 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.

36.4.7.3 USB Endpoint Registers (USEP0–USEP3)

There are four memory-mapped endpoint configuration registers.

	0–3	4–5	6–7	8–9	10	11	12–13	14–15
Field	EPN	—	TM	—	MF	RTE	THS	RHS
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x006C4 (USEP0); 0x006C6 (USEP1); 0x006C8 (USEP2); 0x006CA (USEP3)							

Figure 36-12. USB Endpoint Registers (USEP0–USEP3)

Table 36-11 describes the fields of USEP0–USEP3. The setting for USB host controller should be set only in USEP0, when USMOD[HOST] is set.

Table 36-11. 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
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 can hold only one packet 1 Transmit FIFO can 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 is retransmitted if a transmit error occurs (timeout). RTE can be set only if the transmit packet is contained in a single buffer. If it is not, retransmission should be handled by software. RTE should be cleared for an endpoint configured for ISO transfer mode	Transaction-level interface for USB **host** controller 0 Packet-level interface as described in Section 36.4.1.1, “Packet-Level Interface.” 1 Transaction-level interface as described in Section 36.4.1.2, “Transaction-Level Interface.”

Table 36-11. USEP*n* Field Descriptions (continued)

Bits	Name	USB Function Mode	USB Host Mode
12–13	THS	Transmit handshake for USB **function** controller. This field determines the response to an IN transaction. 00 Normal handshake 01 Ignore IN token 10 Force NACK handshake. Not allowed for control endpoint. 11 Force STALL handshake. On a control endpoint this value is used to generate a protocol stall; the USB function controller clears THS when a SETUP token is received.	Transmit handshake for USB **host** controller 00 Normal handshake
14–15	RHS	Receive handshake for USB **function** controller. This field determines the response to an OUT transaction. 00 Normal handshake 01 Ignore OUT token 10 Force NACK handshake and discard the data. This value can be used for flow control. It is not allowed for a control endpoint. 11 Force STALL handshake. On a control endpoint, this value is used to generate a protocol stall; the USB function controller clears RHS when a SETUP token is received.	Receive handshake for USB **host** controller 00Normal handshake

36.4.7.4 USB Command Register (USCOM)

USCOM is used to start the USB transmit operation.

	0	1	2	3	4	5	6	7
Field	STR	FLUSH	ISFT	DSFT	—		EP	
Reset	0000_0000							
R/W	R/W							
Addr	0x006C2							

Figure 36-13. USB Command Register (USCOM)

Table 36-12 describes the USCOM fields.

Table 36-12. USCOM Fields

Bits	Name	Description
0	STR	Start FIFO fill. Causes the USB controller to start filling the corresponding endpoint transmit FIFO with data. Transmission begins after the IN token for this endpoint is received. The STR bit is always read as cleared.
1	FLUSH	Flush FIFO. Causes the USB controller to flush the corresponding endpoint transmit FIFO. Before flushing the FIFO, the user should issue the Stop_Tx command. After the FIFO is flushed, the user should issue the Restart_Tx command (Refer to Section 36.6, “USB QUICC Engine Commands.”). FLUSH is always read as cleared.
2	ISFT	Increment Start-of-Frame Time. Increments the start-of-frame time by one. ISFT can be used to synchronize the USB frames to an external timing source.

Table 36-12. USCOM Fields (continued)

Bits	Name	Description
3	DSFT	Decrement Start-of-Frame Time. Setting the DSFT bit decrements the start-of-frame time by one. This bit can 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.

36.4.7.5 USB Event Register (USBER)

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 can 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 QUICC Engine module clears the internal interrupt request. This register is cleared at reset.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—				MSF	SFT	RESET	IDLE	TXE4	TXE3	TXE2	TXE1	SOF	BSY	TXB	RXB
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x006D0															

Figure 36-14. USB Event Register (USBER)

Table 36-13 describes USBER fields.

Table 36-13. USBER Fields

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	MSF	Missed start-of-frame. Set when the start-of-frame timer wraps from 11,999 to 0 but no SOF packet has been transmitted because the FIFO was not empty.
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.
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 when the transmit data of the last character in the buffer is 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.

36.4.7.6 USB Mask Register (USBMR)

USBMR is a 16-bit read/write register (0x006D4) in the same bit format as the USBER. If a bit in the USBMR is set, the corresponding interrupt in the USBER is enabled. If the bit is cleared, the corresponding interrupt in the USBER is masked. This register is cleared at reset.

36.4.7.7 USB Status Register (USBS)

USBS, described in Figure 36-15 and Table 36-14, allows the user to monitor real-time status condition on the USB lines.

	0 — 6	7
Field	—	IDLE
Reset	0000_0000	
R/W	R	
Addr	0x006D7	

Figure 36-15. USB Status Register (USBS)

Table 36-14 describes USBS fields.

Table 36-14. 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.

36.4.7.8 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, an SOF packet is transmitted, and USBER[SFT] is set. The USSFT can be read at any time.

	0	1	2 … 15
Field			SFT
Reset	0010_1110_1101_1000		
R/W	R		
Addr	0x006D8		

Figure 36-16. USB Start of Frame Timer (USSFT)

Table 36-15 describes the USSFT fields.

Table 36-15. 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.

36.4.7.9 USB Frame Number (USFRN)

The USFRN value is used as the frame number field in the automatically-transmitted SOF packet in host mode. The USFRN can be read at any time. Normally, it is not necessary to write to this register, but it is permitted if the user wishes to set the frame number.

	0 … 15
Field	FRN
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x006DC

Figure 36-17. USB Frame Number (USFRN)

Table 36-16 describes the USFRN fields.

Table 36-16. USFRN Fields

Bits	Name	Description
0–15	FRN	Frame Number. This field contains the frame number of the current frame. It is automatically incremented every 1 ms.

36.5 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 36-18). These rings have the same basic configuration as those used by the UCCs. There are up to 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 QUICC Engine module confirms reception and transmission or indicates error conditions using the BDs to inform the processor that the buffers have been serviced. The buffers can reside in either external or internal memory.

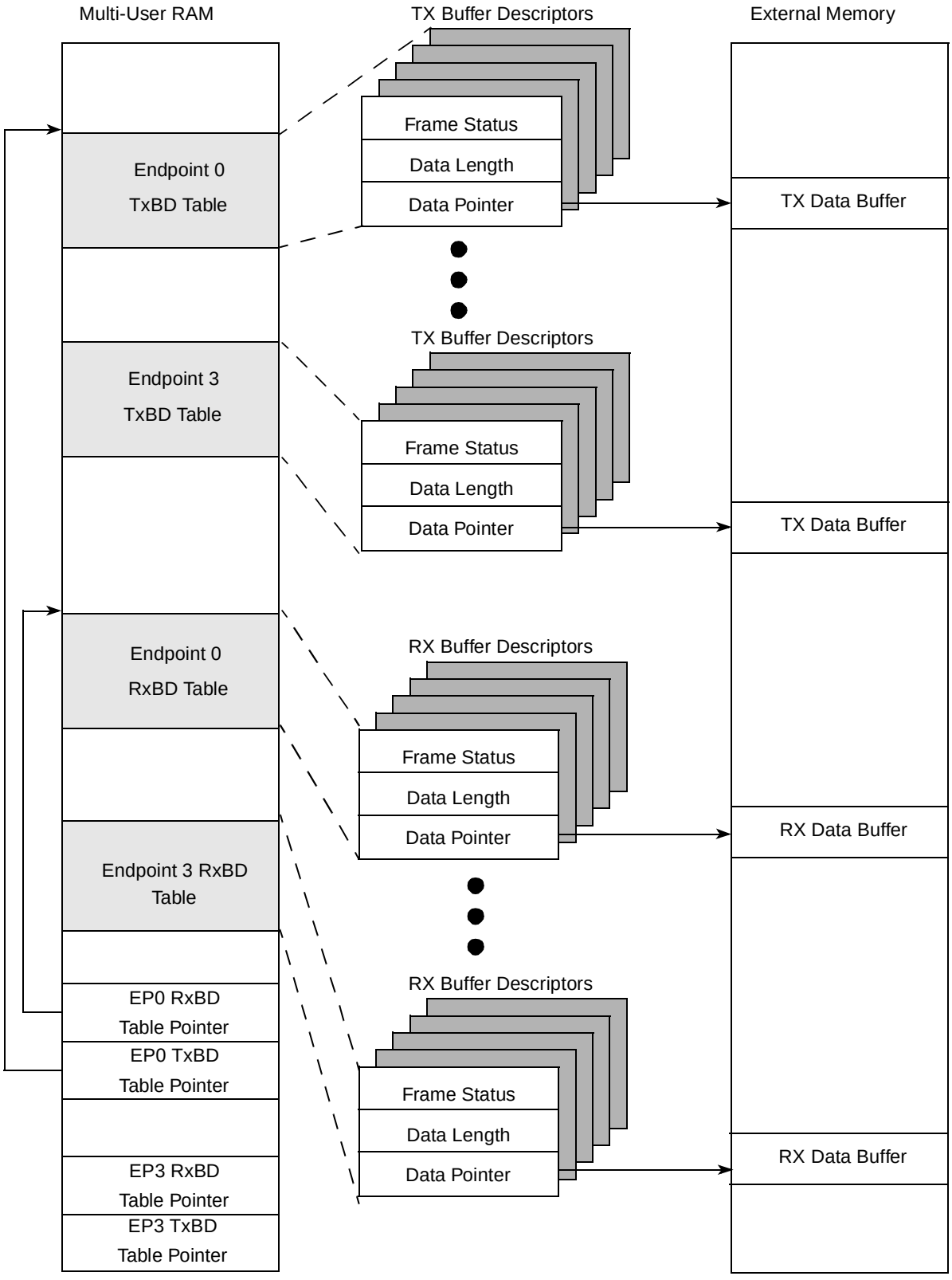


Figure 36-18. USB Memory Structure

36.5.1 USB Receive Buffer Descriptor (RxBD) for Host and Function

The QUICC Engine module reports information about each buffer of received data using RxBDs. The QUICC Engine module 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 36-19, the first word of the RxBD contains status and control bits. The user configures these bits before reception, and the QUICC Engine module sets them after the buffer is 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 the packet-level interface is used) and the function mode. There are no RxBDs in host mode when the transaction-level interface is used.

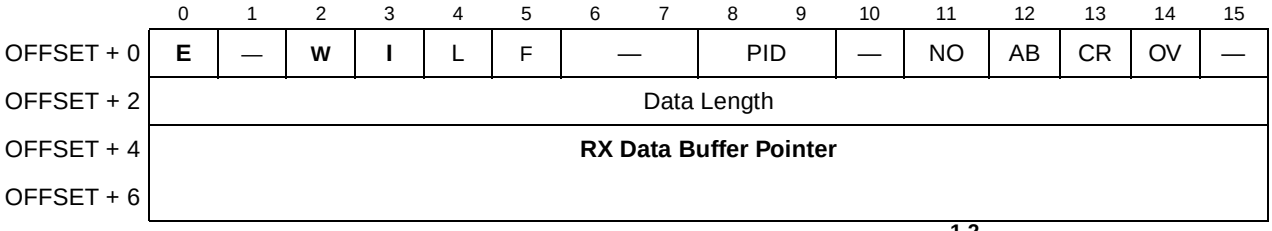


Figure 36-19. USB Receive Buffer Descriptor (RxBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be written by the CPU core before the USB is enabled.

Table 36-17 describes USB receive buffer descriptor fields.

Table 36-17. 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 QUICC Engine module does 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 QUICC Engine module. When 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 QUICC Engine module receives incoming data into the first BD in the table (the BD to which RBASE points). 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 36-17. USB RxBD Fields (continued)

Offset	Bits	Name	Description
0x00	3	**I**	Interrupt 0 No interrupt is generated after this buffer is filled. 1 The USBER[RXB] bit is set when the QUICC Engine module completely fills this buffer, 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. The USB controller sets this bit 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 is 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. The USB controller sets this bit when the buffer contains the first byte of a packet. Written by the USB controller after the received data is 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. The USB controller sets this bit 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 is 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 nonoctet 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 is 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 is 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 is placed into the associated data buffer.
	14	OV	Overrun. A receiver overrun occurred during reception. Written by the USB controller after the received data is 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 QUICC Engine module has written into this BD's data buffer. It is written once by the QUICC Engine module 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 QUICC Engine module has written into this BD's buffer. It is written once by the QUICC Engine module 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 4. The buffer may reside in either internal or external memory.

36.5.2 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.

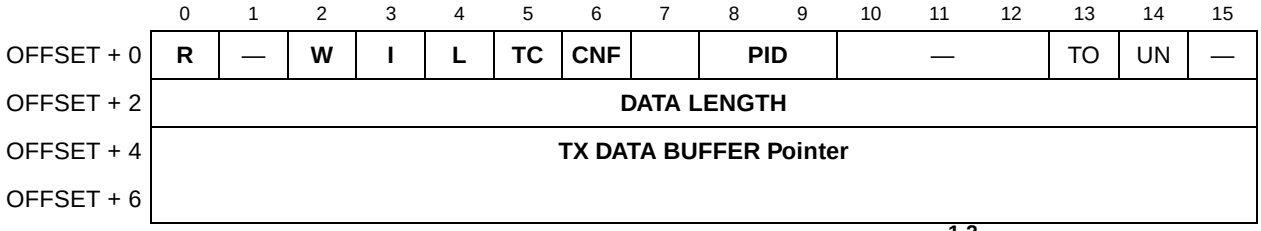


Figure 36-20. USB Transmit Buffer Descriptor (TxBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 36-18 describes USB TxBD fields.

Table 36-18. 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 QUICC Engine module 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 QUICC Engine module sends 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 36-18. USB Function TxBD Fields (continued)

Offset	Bits	Name	Description
0x00	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 36.4.7.3, "USB Endpoint Registers (USEP0–USEP3)." 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 the FIFO. No more packets are loaded to the FIFO after a packet marked CNF until it is 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 QUICC Engine module should transmit from this BD's data buffer. It is never modified by the QUICC Engine module. 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 QUICC Engine module should send from this BD's data buffer. It is never modified by the QUICC Engine module. 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.

36.5.3 USB Transmit Buffer Descriptor (TxBD) for Host

The TxBD described in this section is used when the packet-level interface is active. See Section 36.4.1.1, "Packet-Level Interface."

Data to be transmitted with the USB to the QUICC Engine module by is arranged in buffers referenced by the TxBD ring. The first word of the TxBD contains status and control bits.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**R**	—	**W**	**I**	**L**	**TC**	**CNF**	**LSP**	**PID**		—	NAK	STAL	TO	UN	—
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER Pointer**															
OFFSET + 6																

Figure 36-21. 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 36-18 describes USB TxBD fields.

Table 36-19. 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 QUICC Engine module clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which the user has prepared for transmission, has not been transmitted or is 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 QUICC Engine module sends data using the first BD in the table (the BD to which TBASEx points). 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 is serviced. 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 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 36-19. USB Host TxBD Fields (continued)

Offset	Bits	Name	Description
0x00	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 36.4.7.3, "USB Endpoint Registers (USEP0–USEP3)." 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 QUICC Engine module should transmit from this BD's data buffer. It is never modified by the QUICC Engine module. 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 QUICC Engine module should send from this BD's data buffer. It is never modified by the QUICC Engine module. 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.

36.5.4 USB Transaction Buffer Descriptor (TrBD) for Host

The TrBD described in this section is used when the transaction-level interface is active. See Section 36.4.1.2, "Transaction-Level Interface."

Data to be transmitted with the USB to the QUICC Engine module 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 36-22. 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 36-18 describes USB TrBD fields.

Table 36-20. 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 QUICC Engine module clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which the user has prepared for transmission, has not been transmitted or is being transmitted. The user can write to no fields of this BD after 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 QUICC Engine module sends data using the first BD in the table (the BD to which TBASE points). 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 is 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 Should always have a value of 1 because 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. Should always have a value of 1 to obtain a 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.

Table 36-20. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
0x00	8–9	**PID**	Packet ID. For OUT/SETUP transactions, the user prepares this field 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 nonoctet 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: Timeout. 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 is not modified by the QUICC Engine module. For IN transactions, the user prepares this field with the size of the data buffer, which must be divisible by 4. The QUICC Engine module returns 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 QUICC Engine module sets the BOV bit.
0x04	0–31	**Data Buffer Pointer**	The data buffer pointer. The buffer can 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

Table 36-20. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
0x08	2	—	Reserved, should be cleared.
	3	**ISO**	Isochronous. 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 USEP0[TM] before this transaction is processed.
	5	—	Reserved, should be cleared.
	5–8	**ENDP**	Endpoint. The endpoint number to be included in the token.
	9–15	**ADDR**	Address Device address to be included in the token.
0x0A	0–15	—	Reserved, should be cleared.

Note:

1 Written by the USB controller after it finishes sending or receiving the associated data buffer.

36.6 USB QUICC Engine Commands

The following transmit commands are issued to the QUICC Engine command register (CECR). Refer to Section 19.3.1, “QUICC Engine Command Register (CECR).”

36.6.1 STOP Tx Command

The STOP Tx command disables the transmission of data on the selected endpoint. After the command is issued, the corresponding endpoint FIFO should be flushed. No further data is transmitted until the Restart Tx command is issued.

36.6.2 RESTART Tx Command

The RESTART Tx 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 a transmission error (underrun or timeout).

36.7 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 36-21. Errors that exist exclusively in host mode or function mode are marked as such.

Table 36-21. 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 USBER[TXE] bit. When the packet-level interface is used, the host resumes transmission after reception of the RESTART TRANSMIT command.

Table 36-22 describes the USB controller reception errors.

Table 36-22. 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** transaction-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.

Appendix A
Revision History

This appendix provides a list of the major differences between revisions of the *MPC8323E PowerQUICC II Pro Integrated Communications Processor Reference Manual*.

A.1 Changes From Revision 1 to Revision 2

Major changes to the *MPC8323E PowerQUICC II Pro Integrated Communications Processor Reference Manual*, from Revision 1 to Revision 2 are as follows:

Section, Page	Changes
1.1/1-1	Changed UCC name from “universal communications controller” to “unified communications controller” in the “Five universal communications controllers (UCCs)...” bullet.
1.1/1-1	Changed last bullet of DDR SDRAM memory controller “2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL2 compatible I/O for DDR2” to “2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL_18 compatible I/O for DDR2.”
2.3/2-1	In Table 2-2, “Memory Map,” removed DLLCK from the memory map at location 0x0_1110.
3.1/3-1	Removed extra bullet “QUICC Engine interface signals.”
3.1/3-1	In Figure 3-1, “MPC8323E Signal Groupings,” replaced “RTC_PIT_CLOCK” with “RTC_CLK.”
3.1/3-1	In Table 3-1, “MPC8323E Signal Reference by Functional Block,” and Table 3-2, “MPC8323E Alphabetical Signal Reference,” replaced “RTC_PIT_CLOCK” with “RTC_CLK.”
4.1.2/4-3	In Table 4-2, “External Clock Signals,” added $\overline{\text{CLKIN}}$ and description.
4.3.3.1/4-19	Before Figure 4-5, added the following: “Note that LCS0 is asserted low during the assertion of PORESET and HRESET. Also, PORESET negation to LALE assertion is 36 cycles of CLKIN PCI_SYNC_IN/PCI_CLK clock signal.”
4.3.3.1/4-19	Modified Figure 4-5, “Loading Reset Configuration Words from Local Bus,” and Figure 4-6, “Loading Reset Configuration Words from Local Bus (continued).”
4.4/4-27	Modified Figure 4-9, “Clock Subsystem Block Diagram.”
4.4.2/4-28	Modified “CLKIN” to say “CLKIN/$\overline{\text{CLKIN}}$.”
4.4.3/4-28	Added clarification/stipulation for the formula, *ce_clk* = (primary clock input × CEPMF) ÷ (1 + CEPDF) and added additional formula, to now say:

	"When CLKIN is the primary input clock, *ce_clk* = (primary clock input × CEPMF) ÷ (1 + CEPDF) When PCI_CLK is the primary input clock, *ce_clk* = [primary clock input × CEPMF × (1 + ~CFG_CLKIN_DIV)] ÷ (1 + CEPDF)"
5.3.2.3/5-18	In Table 5-25, "REVID Coding," added revision ID coding for Rev. 1.1 of the device.
5.7.5.4/5-53	Changed introductory sentence for GTCPR*n*.
6.2.6/6-7	In Table 6-7, "AEATR Field Descriptions," added MSTR_ID bit field description.
8.4.2/8-5	In Table 8-2, "IPIC External Signals—Detailed Signal Descriptions," modified the $\overline{\text{IRQ}}$[0:7] signal description.
8.5.3/8-11	In Table 8-10, "SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments," changed field description of bit 10 from reserved to "QE Ports."
9.3.2.2/9-7	In Table 9-4, "Clock Signals—Detailed Signal Descriptions," updated MCKE description.
9.4.1.7/9-18	In Table 9-12, "DDR_SDRAM_CFG Field Descriptions," added a note to bit field 8_BE. In addition, modified the HSE field description.
9.4.1.8/9-21	In Figure 9-9, "DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)," and Table 9-13, "DDR_SDRAM_CFG_2 Field Descriptions," added the DLL_RST_DIS field (bit 2) to DDR_SDRAM_CFG_2, as follows: "DLL reset disable. The DDR controller typically issues a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization. 0: DDR controller issues a DLL reset to the DRAMs when exiting self refresh., 1: DDR controller does not issue a DLL reset to the DRAMs when exiting self refresh."
9.5.6/9-42	Added the following note after the first paragraph:

NOTE

Application system board must assert the reset signal on DDR memory devices until software is able to program the DDR memory controller configuration registers, and must deassert the reset signal on DDR memory devices before DDR_SDRAM_CFG[MEM_EN] is set. This ensures that the DDR memory devices are held in reset until a stable clock is provided and, further, that a stable clock is provided before memory devices are released from reset.

	Added following statement to end of first paragraph: "A gap of 2 dead LCLK cycles is present on the UPM interface between UPM transactions."

10.4.3.2/10-45	Added the following text to end of section: "For proper signalling, the following guidelines must be followed while programming UPM RAM words: • For UPM reads, program UTA and LAST in the same or consecutive RAM words. • For UPM burst reads, program last UTA and LAST in the same or consecutive RAM words. • For UPM writes, program UTA and LAST in the same RAM word. • For UPM burst writes, program last UTA and LAST in the same RAM word."
10.4.3.4.10/10-56	In second paragraph after the third sentence, added the following text: "Setting the WAEN bit for the first RAM word does not have any effect since the first RAM word signifies the start of a new bus cycle and the initial values of the signals driven onto the bus should be present."
11.4.1/11-3	In Table 11-2, "POTARn Field Descriptions," added the following to the TA bit field description: "The translation address must be aligned based on the window's size."
12.3.8.1/12-10	In Table 12-10, "DMAMRn Field Descriptions," modified entire PRC bit field description (bits 11–10). In addition, modified TEM bit description (bit 3).
12.3.8.2/12-12	In Table 12-11, "DMASRn Field Descriptions," modified TE bit description.
13.3.2.11/13-23	Added the following to the end of the first paragraph: "Inbound and outbound windows for the same bus should not overlap. Therefore, situations where an inbound window translation points back into an outbound window, or where an outbound translation window points back into an inbound window, are not allowed."
13.3.2.11/13-23	In Table 13-18, "PITARn Field Descriptions," added "The specified address must be aligned to the window size, as defined by PIWAR*n*[IWS]" to the description of TA.
13.3.2.12/13-24	In Table 13-19, "PIBARn Field Descriptions," added "The specified address must be aligned to the window size, as defined by PIWAR*n*[IWS]" to the description of BA.
19.4.4/19-19	Split Table 19-7, "CEVTER/CEVTMR Field Descriptions," into two tables, Table 19-7, "CEVTER Field Descriptions," and Table 19-8, "CEVTMR Field Descriptions."
20.2/20-3	Added references to new sections, as follows: "See details on the CMXSI*nXXX* registers: • Section 20.5.2, "CMX SI1 Clock Route Low Register (CMXSI1CRL)" • Section 20.5.3, "CMX SI1 SYNC Route Register (CMXSI1SYR)"

	• Section 20.5.4, “CMX UCC Clock Route Register (CMXUCR1)’”
20.3/20-3	Modified Figure 20-2, “Enabling Connections to the TSA.”
21.9.2/21-26	In Figure 21-17, “SPI TxBD,” and Table 21-13, “SPI TxBD Status and Control Field Descriptions,” changed SPMODE[18] from Reserved to MII.
22.3.2/22-5	Changed the sentence, “Note that while the UCC receiver and transmitter functions can be set independently, they are set to the same mode (that is, full duplex fast or slow) in most applications,” to the following: “Note that while it is possible to program the UCC receiver and transmitter functions independently, they should always be set to the same configuration (that is, for fast or slow protocols).”
22.3.2/22-5	In Table 22-4, “GUEMR Field Descriptions,” changed the note in URMODE bit description to, “**Note:** URMODE should be programmed to have the same value as the UTMODE bit.”
24.1/24-1	Modified third sentence to read, “UART links are character-based.”
Chapter 26/26-1	Replaced “EFM” with “HDLC” and replaced “100 Mbps” with “70 Mbps” in the first bullet.
Chapter 26/26-1	Replaced the seventh bullet “Data buffers” with the following: “Buffers and buffer descriptors (BDs) may reside anywhere in system memory.”
Chapter 26/26-1	Added the following bullet as the first bullet: • Supports the following interfaces: — UTOPIA L2 at 50 MHz (assuming QUICC Engine frequency above 100 MHz[1]). — MII/RMII (assuming QUICC Engine frequency above 100 MHz). — High speed serial interface[2] with modem control. — Serial or nibble-parallel data interface through TDM[3]. The notes are as follows: 1. Note that this is the minimal QUICC Engine frequency for H/W operation. The actual QUICC Engine frequency to meet the protocol performance target will be higher. 2. The QUICC Engine frequency must be higher than the data bit-rate multiplied by 8/3. 3. The QUICC Engine frequency must be higher than the data bit-rate multiplied by 24.
26.3/26-4	Modified the first paragraph of the note, as follows: **NOTE: Backward-Compatibility** After power-on reset, UCC1–UCC3 are backward-compatible to FCC1–FCC.

26.4.2.1/26-6	In Table 26-2, “GUMR (in Fast Mode) Register Field Descriptions,” removed the following note in the DIAG field description: “**Note:** In Ethernet mode program MACCFG1[8] bit for loopback mode. DIAG bits have no effect.” In addition, in the definition for the ENR bit, changed “MACCFG1 bit 2 must be set,” to “MACCFG1 bit 29 must be set,” and changed “MACCFG1 bit 0 must be set,” to “MACCFG1 bit 31 must be set.” In addition, modified note in CPD field descriptions.
26.5.3/26-14	In Table 26-6, “URFET (in Fast Mode) Register Field Descriptions,” changed “TRSH” to “RFET.”
26.5.4/26-14	In Table 26-7, “URFSET (in Fast Mode) Register Field Descriptions,” changed “STRSH” to “RSFET.”
26.5.8/26-16	In Table 26-11, “UTFTT (in Fast Mode) Register Field Descriptions,” modified TFTT description.
29.1/29-1	Modified Figure 29-2, “VLAN Tagged Ethernet Frame Structure.”
29.4.7/29-14	Changed “Programmable Parse Command Descriptor (PCD) Mode” to “Extended parsing mode.”
29.4.7.1/29-14	In Figure 29-6, “Address Filtering Flow REMODER[EXF]=0,” changed “Extended Filtering Mode” to “Extended Parsing Mode.”
29.4.7.1.1/29-16	In the second paragraph, modified sentence, as follows: “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.”
29.4.7.2.3/29-18	Modified Figure 29-8, “Address Filtering Flow REMODER[EXF]=1,” by changing first decision block from UPSMR[EXP]=1 to REMODER[EXP]=1. Added False branch and cross reference.
29.4.8/29-19	In second paragraph, changed “REMODER[VNoTagOP]” to “REMODER[VNonTagOP].”
29.4.10/29-19	In Figure 29-9, “Receive Queueing Decision,” changed bottom half of center box to read, “If REMODER[EXP] = 1, by TCI in Termination Action Descriptor (TAD).”
29.5.2.1/29-46	In Table 29-24, “Transmit Buffer Descriptor Field Description,” replaced “VTBL_BASE” with “VTAGTable” in the RC/VID field of the transmit buffer descriptor.
29.5.3/29-53	Removed everything under the sentence “The Ethernet INIT command must be executed after Rx and Tx Parameter RAM have been initialized by the user.”
29.5.3.1/29-54	In Figure 29-34, “Transmitter Parameter RAM Data Structures,” switched the order of SQPTR and TQPTR to reflect the true order in the Parameter RAM. In addition, replaced “Size = 136..416 Bytes” with “Size = 136 × Num_of_Threads bytes” and replaced “Size = 64..512 Bytes” with “Size = 64 × Num_of_Queues bytes”.

29.5.3.3/29-56	In Table 29-29, “Scheduler Programming Model,” added the following sentence to the description for the SQPTR field: “This address must be 32 bytes aligned.”
29.5.3.3.1/29-57	Added the following sentence at the end of the paragraph: “The user must allocate 136 bytes per thread.”
29.5.3.3.3/29-58	In Table 29-32, “RX Global Parameter RAM,” added the following to the description of the FracSiz field: “See example calculations in Section 29.15, “Traffic Shaper Programming Considerations.” Note: CPU must write a complete byte when Initializing this value or changing it on the fly (3 upper bits must be zero) although only the 5 lowest bits are actually used to determine the FracSiz value (FracSize value is between 0 to 31).The QE may change the unused bits (3 upper bits) on run time.” In addition, added the CPU-initialized 8-bit SCHSTATR field to offset 0x53 with the following description: “Scheduler Status register. See Section 29.5.3.3.4, “Scheduler Status Register (SCHSTATR).”
29.5.3.3.4/29-61	Added Section 29.5.3.3.4, “Scheduler Status Register (SCHSTATR).”
29.5.3.7/29-65	In Table 29-36, “Static Parameter (SP) Field Description,” changed VNonTagOP field description to “A Q tag is inserted between the SA and the incoming T/L field.”
29.5.3.10/29-71	Modified Table 29-37, “RxBD Parameter Table Description.”
29.5.3.11/29-71	Added the following sentence at the end of the paragraph: “The incoming frame is placed in the priority queue as programmed in the PRx field of the the L2QT or L3QT register.”
29.5.3.11.2/29-72	Modified Table 29-39, “L3QT Description.”
29.5.5/29-74	In Table 29-41, “BMRx Field Descriptions” changed decimal values from hexadecimal to decimal: “0x0” to “0”, “0x1” to “1”, “0x62” to “62”, “0x63” to “63”.
29.6/29-75	Changed third sentence to read, “The Global Parameter RAM 16 byte EXPGlobalParam is programmed to point to the first Parse Command Descriptor (PCD).”
29.6.2.6.5/29-87	In Table 29-54, “RX Firmware Counters,” added “These bits are used to determine the Rx BD Ring for the CPU” after “RQoS Mode” in the HLUT RQoS field description.
29.7.4.1/29-91	In Table 29-55, “UCC Statistics,” removed “e.g. OV/UN (30.1.1.12).”
29.8.1/29-107	Modified second paragraph to read as follows: “The INIT RX PARAMETERS Command is issued in order to initialize the Receive Hardware, and some internal parameters.”
29.8.2.1/29-115	In Table 29-77, “Source Port Values (SNUMs),” modified bit field “Rx Thread1 Parameter RAM Page” description.

29.8.2.1/29-115	Changed “GenerateL2 LookupKey PCD” to “GenerateLookupKey_EthFast PCD.”
29.8.2.1/29-115	In Table 29-79, “Signal Descriptions,” added UCC1 (SNUM 0x01) and UCC5 (SNUM 0x41).
29.12/29-123	In Table 29-84, “Rx Parameter RAM Usage,” added “8” to the range for the TxT and RxtT parameters.
29.12/29-123	In Table 29-85, “Ethernet Multiuser RAM Usage,” changed the Thread Data Structure’s Gigabit Ethernet Example value total to 544. In addition, changed resulting Total Tx Parameter RAM Usage from Gigabit Ethernet Example to 1344.
29.12/29-123	In Table 29-86, “Ethernet No LLC Header Format,” changed data structure name “ThreadQ” to “Thread Data Structure.” In addition, added footnote for “Thread parameter RAM.”
30.3.6.1/30-82	Modified last sentence of second paragraph to read, “For ATM MPHY Slave mode, the APC PHY parameter table is located only on the Last PHY location.”
31.3.2/31-7	Added “Section 31.3.2, “Transmit Internal/External Rate Modes.”
32.3/32-6	Changed last bullet in feature list.
34.3.3/34-12	Removed Figure 34-8, ”Time Slot Assignment Tables for 256 Channels over 4 UCCs.”
34.6.2/34-36	In Table 34-15, “Transmit Buffer Descriptor (TxBD) Field Descriptions,” modified bit 4 “L” field description.
Chapter 36	Removed Chapter “EFMC Controller.”
36.1.1/36-1	Removed the statement “Note that low-speed operation requires an external hub.” from the end of this bullet in the second set of bulleted text— “Supports both 12- and 1.5-Mbps data rates (automatic generation of preamble token and data rate configuration).”

A.2 Changes From Revision 0 to Revision 1

Major changes to the *MPC8323E PowerQUICC II Pro Integrated Communications Processor Reference Manual*, from Revision 0 to Revision 1 are as follows:

Section, Page	Changes
Preface	PCI chapter description modified to complies with PCI Local Bus Specification, 2.3.
1.2.1/1-6	Revised to state that caches use PLRU, TLBs use LRU Corrected to state completion and retirement of 2 instructions per cycle Listed XOR under ASEU Moved AFEU out to be aligned with other EUs
Table 2-3, Table 2-4	Changes to QE memory map:

	Changed MPHY1/2 to UCC1/2
	Hid rows for Debug, RISC1 SPCL, RISC2 SPCL, Test
	Added 1588 registers at 0x0_4800
2.3/2-1	Added three paragraphs about reading and writing to registers and reserved bits. (“Reading from address locations... The description of the specific bits will indicate when this is needed.”)
3.3/3-12	In Table 3-5, hid SICRH reference
4.3.2.2/4-14	Reorganized and repaired sections relating to RCWLR and RCWHR and their fields. (Starting with Section 4.3.2.1, “Reset Configuration Word Low Register (RCWLR),” and ending with Section 4.3.3.1, “Loading from Local Bus EEPROM.”) Removed duplication.
4.3.3.2.2/4-22	In Table 4-55, bit range of fourth row from bottom changed to 20-27.
Table 4-9	Changed “Reserved, should be cleared” to “Reserved, should not be set” for SMPF values of 0000, 0111, and 1111 because the chip will malfunction if clkdiv is asserted and SMPF holds a value of 0.
Table 4-5	Corrected valid frequency range for PCI_CLK/PCI_SYNC_IN from 25–66.666 MHz to 24–66.666 MHz
Table 4-9	Changed description for SPMF = 0000 from “Reserved, should not be set” to “Reserved”
5.4.4.12/5-121	Removed references to MDIC0 and MDIC1
	Removed references to DHC_EN, MDIC0_OE, and MDIC1_OE
	Removed all text from “Note that the MDIC signals..” to “1111”
5.3.2.3/5-18	Removed SPRIDR[PARTID] values and replaced them with a reference to (already existing) Table 5-24.
5.2.4.1.1/5-6	Added a bullet with sub-bullets regarding the e300 core writing to the IMMRBAR
5.3.2.5/5-21	In Table 5-27, swapped bits 16 and 17 and swapped SICRL[16] and SICRL[17] so that SICRL[17] controls GPIO1[9] and SICRL[16] controls GPIO1[10].
5.4.4.1/5-25	Table 5-29, changed bit 31 from “divide-by-8,192” to “divide-by-65,536”
5.5.6.1/5-35	Replaced “external 32.768 kHz crystal” with “external clock source”
5.3.2.6/5-22	In Figure 5-41 and Table 5-96 corrected DDRCDR figure to show DDR_cfg (table was showing bit 13 as DDR_cfg, figure showed it as reserved)
7.3.7/7-35	Removed paragraph describing bus arbitration scheme
Chapter 7/7-1	Removed references to a moded 32- or 64-bit data bus because that is a feature that has not been integrated into the SoCs and it will be phased out of e300 cores. The default now is a 64-bit data bus
8.5.3/8-11	In Table 8-10, made bit 10 reserved
Figure 8-1	In Figure 8-1, changed number of signals between the QE and the IPIC from 3 to 2.

Table 8-12	Changed ipi_int_internal[7] from IDMA to Reserved
Chapter 8/8-1	Throughout chapter, corrected references to INTA and MCP_OUT to show as active-low ($\overline{\text{INTA}}$ and $\overline{\text{MCP\_OUT}}$)
8.5.19/8-27	Corrected offsets:
8.5.20/8-27	SCVCR from 0x50 to 0x60 SMVCR from 0x54 to 0x64
8.1/8-1, 8.2/8-4	Changed 3 internal signals to 1 external and 2 internal signals that cause an $\overline{mcp}$ interrupt.
8.1/8-1	Updated the section (towards the end) by removing the specific counts of each type of signal to read: "The IPIC receives the following types of interrupts: —External interrupt—triggered by the off-chip signals ($\overline{\text{IRQ}}n$) listed in Table 5-32. —Internal interrupts—on-chip interrupts, triggered by the sources listed in Table 8-8 and Table 8-10 —External and internal non-maskable machine check conditions, signaled by the sources listed in Table 8-22 through $\overline{mcp}$"
9.4.1.16/9-28	Removed DDR_INIT_EXT_ADDR (section number and page refer to Rev. 0)
Figure 9-1, Table 5-32	
Table 4-1, Figure 9-19	Removed $\overline{\text{MDQS}}$[0:8]
9.2/9-2, 9.5.1/9-32	Corrected maximum SDRAM device density for to be 1 Gbit, 2 Gbit for some internal device configurations
10.3.1.2.1/10-10	Replaced Table 10-5.
13.3.1/13-12	Added the following sentence to the end of the first paragraph: "The PCI registers, PCI_CONFIG_ADDRESS, PCI_CONFIG_DATA, and PCI_INT_ACK, are little endian registers."
Figure 13-43	Split up PRI*n* into PRI0, PRI1, etc. to avoid confusion on which end PRI0 begins.
15.1.2/15-2	Revised first two bullets as follows —Changed first bullet from "The I^2C is the driver of the SDA line." to "The I^2C initiates a transfer, generates clock signals, and terminates a transfer." —Changed second bullet from "The I^2C is not the driver of the SDA line." to "The I^2C is addressed by an I^2C master."
16.2.2/16-3	In Table 16-1, added UART_SIN*n*, UART_SOUT*n*, $\overline{\text{UART\_CTS}}n$, and $\overline{\text{UART\_RTS}}n$ signals.
17.2/17-1	Added word 'all' to first sentence after bulleted list: —The TDI and TDO signals input and output instructions and data to the JTAG scan registers. now reads:

	—‘....input and output all instructions....’ Changed following sentence from: —[old] The boundary-scan operations are controlled by the TAP controller through commands received by means of the TMS signal. to: —[new] JTAG operations are controlled by the TAP controller through the TMS and TCK signals.
19.3.1.1/19-5	In Table 19-5, added missing row GRACEFUL STOP RECEIVE with opcode 011010.
24.18.1/24-33	In the UCCE/UCCM, shown in Figure 24-17 and Table 24-22, corrected bits 9, 10, 11 to match in both the figure and the table.
25.3.1/25-4	In Table 25-3, changed the offset for UPSMR from 0x04 to 0x08. Added EFM to features list.
26.4.6/26-11	Added the Bus Mode Registers and the cross reference to them from the UCC Transparent and HDLC chapters.
28.4.2.1/28-4	Removed verbiage about FCRx’s, which are correctly referred to as Bus Mode Registers. Added the Bus Mode Registers to the UCC Fast Protocols chapter, and updated the cross-references in Table 28-2 to correctly point to them.
27.3.3/27-17	In section, where stated, “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,” changed to, “If the echo bit is ever 0 when the transmit bit is 1, a collision occurs between terminals; the station(s) that sent a one stops transmitting. The station that sent a zero continues as normal.”
29.5.1.8/29-38	In Table 29-15, changed PHY address field description, with additional information on reprogramming TBIPA[TBIPA].
30.3.2.7/30-60	Added a note in the address lookup, at VCLT_SF bit, Table 30-26, allowing for larger scale factor calculations.
Figure 30-54/30-101	In the second paragraph below Figure 30-54, describing the WFQ table, removed the statement requiring that the table be 64-byte aligned. Added further explanation of memory allocation if the table is located in external memory.
32.6.1/32-10	In Table 32-4, removed the wording “and UPUC[LB] from the footnote.
Chapter 32	Removed references to 50MHz.
Chapter 34	Clarified information in the overview and features sections. In Section 34.3.3, “Multiple UCC Assignment Tables,” added a third scenario and Figure 34-8. Corrected the UCC assignment designators in Figure 34-6.
Chapter 36	Added “EFMC Controller” chapter.
36.4.5/36-15	Rewrote section to more clearly explain function mode. Also added a figure and table of description for host mode.
36.4.7.9/36-22	In section, added that this is for Host Mode.

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 PowerPC architecture 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 e300 bus interface that may extend across multiple bus cycles. A e300 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 PowerPC architecture. 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 PowerPC 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

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.

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.

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

IEEE 754. A standard written by the Institute of Electrical and Electronics Engineers that defines operations and representations of binary floating-point numbers.

Illegal instructions. A class of instructions that are not implemented for a particular PowerPC processor. These include instructions not defined by the PowerPC 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 PowerPC architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features. The PowerPC architecture has many different implementations.

Inbound 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.

Inter-packet gap. The gap between the end of one Ethernet packet and the beginning of the next transmitted packet.

Integer unit. An execution unit in the core responsible for executing integer instructions.

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. See *Out-of-order.*

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

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that execution for a subsequent instruction.

L2 cache. Level-2 cache. See *Secondary cache*.

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.

Medium-dependent interface (MDI) sublayer—Sublayer that defines different connector types for different physical media and PMD devices.

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.

MEI (modified/exclusive/invalid). *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.

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 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

Outbound Windows. Mappings that perform address translations from local 32-bit address space to the address spaces of PCI1 or PCI2, which may be much larger than the local space. Outbound 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.* On PowerPC processors, 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.

Precise exceptions. A category of exception for which the pipeline can be stopped so instructions that preceded the faulting instruction can complete and subsequent instructions can be flushed and redispatched after exception handling has completed. See *Imprecise exceptions.*

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.

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.

RISC (reduced instruction set computing). An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

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* and *Execution synchronization.*

Synchronous exception. An *exception* that is generated by the execution of a particular instruction or instruction sequence. There are two types of synchronous exceptions, *precise* and *imprecise*.

System memory. The physical memory available to a processor.

T

Time-division multiplex (TDM). A single serial channel used by several channels taking turns.

Tenure. The period of bus mastership. For the e300, there can be separate address bus tenures and data bus tenures. A tenure consists of three phases: arbitration, transfer, and termination.

TLB (translation lookaside buffer). A cache that holds recently-used *page table entries*.

Throughput. The measure of the number of instructions that are processed per clock cycle.

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.

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

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

Z